SIMULASI PENCEGAHAN *DEADLOCK* PADA KASUS *PRODUCER-CONSUMER PROBLEM* SEBAGAI ILUSTRASI DALAM SISTEM OPERASI

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

Oleh:

THORIQSON OPERA NIRWANA
10451025569

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2011

SIMULASI PENCEGAHAN *DEADLOCK* PADA KASUS *PRODUCER-CONSUMER PROBLEM* SEBAGAI ILUSTRASI DALAM SISTEM OPERASI

THORIQSON OPERA NIRWANA

10451025569

Tanggal Sidang : 24 Juni 2011
Periode Wisuda : November 2011

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau
Jl.Soebrantas KM 15 No.155 Pekanbaru

ABSTRAK

Sistem operasi memiliki tugas untuk mengatur proses-proses yang berjalan dalam jangka waktu yang sama tanpa boleh saling bertabrakan satu dengan yang lainnya. Proses-proses yang berinteraksi memerlukan *sinkronisasi* agar terkendali dan terhindar dari kondisi *deadlock*. Kasus *producer-consumer problem* digunakan sebagai ilustrasi pembahasan *sinkronisasi* pada sistem operasi, yaitu dua proses saling berbagi *buffer* dengan jumlah yang terbatas. *Deadlock* pada kasus ini adalah dua proses menjadi *blocked,* dalam hal ini masing-masing proses menjadi *blocked* karena menunggu komunikasi dari proses lain.

Tugas akhir ini di rancang untuk mengilustrasikan dan menvisualisasikan kasus *producer-consumer problem* dalam bentuk simulasi. Simulasi adalah suatu teknik meniru operasi atau proses yang terjadi dalam suatu sistem dengan bantuan perangkat komputer dan dilandasi oleh beberapa asumsi tertentu sehingga sistem tersebut bisa dipelajari secara ilmiah. Simulasi *producer-consumer problem* yang di buat menerapkan metode *sleep and wake-up* agar proses-proses yang berjalan dapat terhindar dari kondisi *deadlock*. Simulasi ini dapat digunakan sebagai salah satu alternatif untuk membantu Pemahaman tentang kasus *Producer-consumer problem*.

Kata kunci : *Buffer, Blocked, Deadlock, Interface, Producer-Consumer Problem, Proses, Sleep and Wake-up, Sinkronisasi.*

SIMULATION PREVENTION DEADLOCK STUDY IN PRODUCER-CONSUMER PROBLEM IN THE OPERATING SYSTEM ILLUSTRATION

THORIQSON OPERA NIRWANA

10451025569

Date of Final Exam : Juny 28th 2011
Date of Graduation : No 2011

Informatics Engineering Departement
Faculty of Sciences and Technology
State Islamic University of Sultan Syarif Kasim Riau

The operating system has a duty to manage the processes running in the same period may collide with each other without one another. Interacting processes that require synchronization in order to control and avoid the deadlock condition. The case of producer-consumer problem used to illustrate the discussion of synchronization in operating systems, namely the two processes share the buffer with a limited number. Deadlock in this case is two processes to be Blocked, in this case each process becomes blocked waiting for communication from another process.

The final task is designed to illustrate and visualize the case of producer-consumer problem in the form of simulation. Simulation is a technique mimics the operation or process that occurs in a system with the help of computer equipment and based on some specific assumptions so that the system can be studied scientifically. Simulation of producer-consumer problem in applying the method for sleep and wake-up so that the processes that are running can avoid deadlock conditions. These simulations can be used as an alternative to aid understanding of the cases Producer-consumer problem.

Key words: buffer, Blocked, Deadlock, Interface, Producer-Consumer Problem, Process, Sleep and Wake-up, Sync.

DAFTAR ISI

Halaman

DAFTAR ISTILAH

Aplikasi	suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna
Assembler	Sebuah program komputer untuk menterjemahkan Bahasa Assembly (bahasa pemrograman komputer tingkat rendah)
Blocked	yaitu status dimana proses tidak dapat dijalankan pada saat prosesor siap/bebas, Status yang dimiliki pada saat proses menunggu suatu sebuah event seperti proses I/O**.**
Buffer	Area memori yang menyimpan data sementara ketika mereka sedang dipindahkan antara dua device atau antara device dan aplikasi
Compiler	Program komputer yang berguna untuk menerjemahkan program komputer yang ditulis dalam bahasa pemrograman tertentu menjadi program yang ditulis dalam bahasa pemrograman lain.
CPU	Suatu perangkat keras microprocessor yang memahami dan melaksanakan suatu perintah dari perangkat lunak
Critical section	Bagian dari proses yang memerlukan *mutual exclusion*
Efisien	Penggunaan sumber daya secara minimum guna pencapaian hasil yang optimum.
Form	Bentuk dari sebuah tampilan
Hardware	Sebuah alat/benda yang kita bisa lihat, sentuh, pegang dan memiliki fungsi tertentu
Implementasi	Pelaksanaan atau penerapan
Input	Data yang dimasukkan
Interface	Tampilan antar muka
I/O *device*	*B*agian dari sistem mikroprosesor yang digunakan oleh mikroprosesor itu untuk berhubungan dengan dunia luar

Komponen	Bagian dari keseluruhan atau unsure
Memori	Istilah generik bagi tempat penyimpanan data dalam komputer
Multithreading	Sistem yang memungkinkan lebih dari satu *thread* dieksekusi secara bersamaan
Mutual Exclusion	Adalah jaminan hanya satu proses yang mengakses sumber daya pada suatu interval waktu tertentu.
Output	Data yang dihasilkan
Program	adalah kumpulan instruksi atau perintah yang disusun sedemikian rupa sehingga mempunyai urutan nalar yang tepat untuk menyelesaikan suatu masalah.
Proses	Unit kerja terkecil yang secara individu memiliki sumber daya-sumber daya dan dijadwalkan sistem operasi
Program counter	Register yang bertugas untuk mencatat alamat memori dimana instruksi di yang akan eksekusi.
***Process Control Block* (PCB)**	Manifestasi dari suatu proses dalam suatu sistem operasi
Thread	Abstraksi dari unit aktivitas (penjadwalan)
Race condition	Situasi dimana beberapa proses mengakses dan memanipulasi data secara bersamaan
RAM	Sebuah tipe penyimpanan komputer yang isinya dapat diakses dalam waktu yang tetap tidak memperdulikan letak data tersebut dalam memori
Resource	Sumber daya-sumber daya dan dijadwalkan sistem operasi
Software	Sekumpulan data elektronik yang disimpan dan diatur oleh computer
Sinkronisasi	Proses pengaturan jalannya beberapa proses pada saat yang bersamaan

Sistem operasi	merupakan suatu program yang bertindak sebagai *interface* antara user dan sistem komputer. Sistem operasi memiliki tugas untuk mengatur proses-proses yang berjalan dalam jangka waktu yang sama tanpa boleh saling bertabrakan satu dengan yang lainnya
Simulasi	Proses merancang model dari suatu sistem yang sebenarnya
User	Pemakai

DAFTAR TABEL

BAB IV
ANALISA DAN PERANCANGAN

4.1 Analisa

Dalam subbab ini, akan dibahas mengenai alur perangkat lunak simulasi, penggambaran objek simulasi dan proses dari Simulasi *Producer-Consumer Problem*. Masing-masing pembahasan akan dibahas dalam beberapa sub bab berikut ini.

4.1.1 Analisa Permasalahan

Kasus *producer-consumer Problem* digunakan sebagai ilustrasi pembahasan sinkronisasi. Masalah *producer-consumer* disebut juga *bounded-buffer problem* (masalah *buffer* dengan jumlah terbatas).

Asumsi dalam *producer-consumer problem* adalah sebagai berikut, Dua proses, menggunakan suatu *buffer* yang dipakai bersama dan berukuran terbatas. Satu proses adalah *producer* yang meletakkan informasi ke *buffer*, Proses lain adalah *consumer* yang mengambil informasi dari *buffer*.

Karena *buffer* terbatas, masalah berikut dapat terjadi, yaitu:

1. Masalah untuk *producer*.

   Masalah terjadi ketika *buffer* telah penuh, sementara *producer* ingin meletakkan informasi ke *buffer* yang telah penuh itu.

2. Masalah untuk *consumer*.

   Masalah terjadi ketika *consumer* ingin mengambil informasi sementara *buffer* telah/sedang kosong.

Deadlock pada kasus ini adalah dua proses menjadi *Blocked,* dalam hal ini masing-masing proses menjadi *Blocked* karena menunggu komunikasi dari proses lain. *Producer* dan *Consumer* saling menunggu komunikasi dari lainnya dan tidak dapat beranjak dari kondisi ini. Kedua proses memerlukan sinkronisasi agar keduanya terhindar dari kondisi *Deadlock.*

4.1.2 Analisa Metode

Masalah ini dapat di dihindari dengan menggunakan metode *Sleep and Wake-up.* Penyelesaian ini memiliki dua rutin, yaitu *sleep* dan *wakeup*

1. *Sleep* adalah *system call* membuat proses yang memanggil di blok *(blocked).*
2. *Wake up* adalah *system call* yang membuat proses yang memanggil menjadi *ready.*

Mekanismenya proses akan di blok/tidur *(sleep)* apabila tidak bisa memasuki *critical section*-nya dan akan dibangunkan *(wake up)*/*ready* apabila resource yang diperlukan telah tersedia. Kedua rutin bersifat *atomic,* yaitu saat rutin dieksekusi maka tak ada interupsi yang dapat menyela.

Solusi penyelesaian dengan *sleep and wakeup* adalah sebagai berikut:

1. Solusi untuk masalah *producer*.

Producer memanggil *sleep* begitu mengetahui *buffer* telah penuh saat *producer* akan menyimpan informasi ke *buffer*. *Producer* tidak lagi aktif kecuali dibangunkan (*wake-up*), proses lain (*consumer*) yang memberitahu bahwa satu *barang* atau lebih telah diambil dari *buffer* sehingga terdapat ruang bagi *producer* untuk menyimpan informasi ke *buffer***.**

2. Solusi untuk masalah *consumer*.

Consumer memanggil *sleep* begitu mengetahui *buffer* telah kosong saat *consumer* mengambil *barang*. *Consumer* tidak lagi aktif kecuali dibangunkan (*wakeup*) proses lain (*producer*) yang memberitahu bahwa *buffer* telah terisi satu *barang* atau lebih sehingga terdapat informasi yang dapat diambil *consumer* dari *buffer*

4.1.3 Analisa simulasi

Kasus *producer-consumer problem* di modelkan sebagai sebuah pasar, dimana terdapat *producer, consumer* dan *market*. *Producer* adalah variable yang memasukkan barang ke *market*, sementara *consumer* adalah variable yang membeli barang dari *market*. Analoginya dengan sistem operasi adalah sebagai berikut:

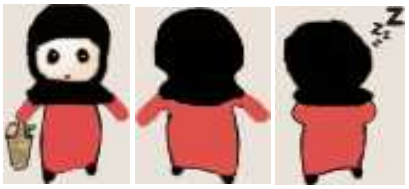
1. Producer

Producer di analogikan sebagai proses yang meletakkan informasi ke *buffer*

Gambar 4.1 Gambar *producer*

2. Consumer

Consumer di analogikan sebagai proses yang menghapus informasi dari *buffer*



Gambar 4.2 Gambar *Consumer*

3. Market

market di analogikan sebagai *buffer*, sumber daya dengan jumlah terbatas yang di akses oleh beberapa proses. Jenis barang pada *market* mewakili jumlah slot yang ada pada *buffer*, sedangkan angka pada *buffer* di analogikan sebagai informasi yang tersimpan di dalam *buffer*

Gambar 4.3 Gambar *Buffer*

4. Gambar pelengkap lainnya, seperti: gambar rumah *producer* dan *consumer*, dan beberapa gambar lainnya.

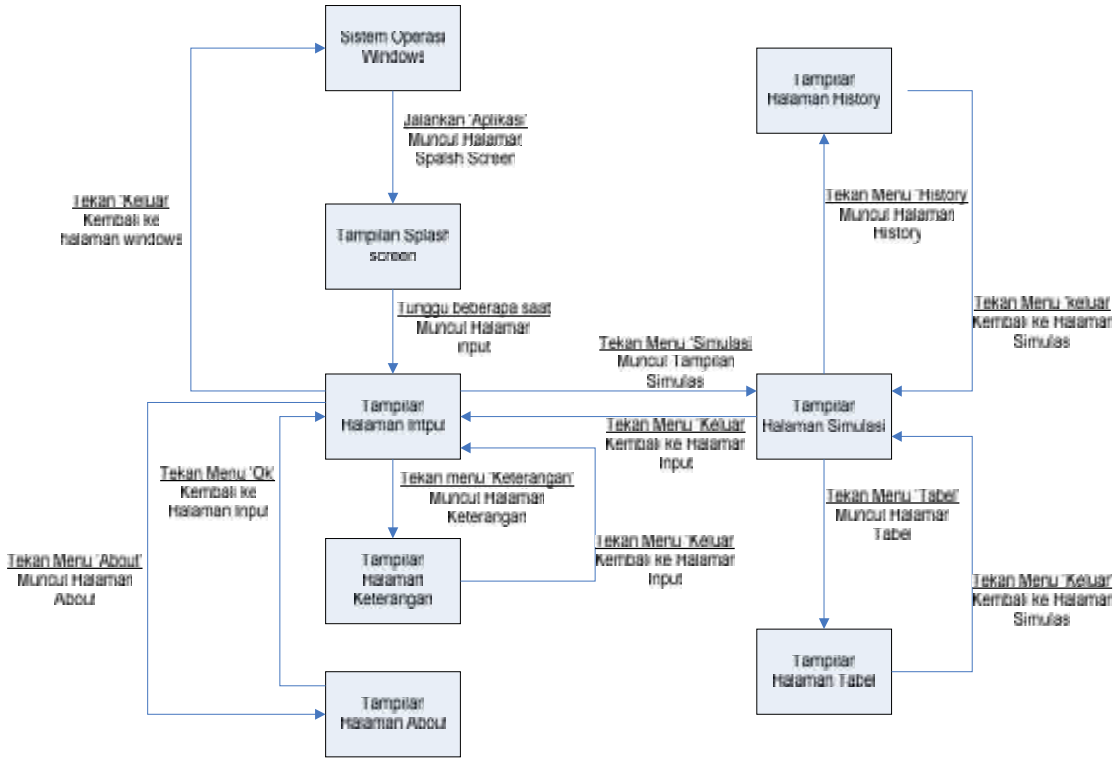
4.2 Perancangan

4.2.1 Perancangan Alur kerja Simulasi

Simulasi *Producer-Consumer Problem* dimulai dengan tampilan *splash screen*. *Halaman splash screen* berisi nama/judul perangkat lunak dan nama pembuat perangkat lunak. Beberapa saat kemudian, *form input* akan tampil. *Form input* berfungsi untuk mengatur kondisi simulasi. Komponen yang dapat diatur adalah sebagai berikut:

1. Pengaturan pada *Producer*, yaitu jumlah *producer*, batas maksimum dan minimum bagi *producer* dalam satu kali produksi.
2. Pengaturan pada *Consumer*, yaitu jumlah *consumer*, batas maksimum dan minimum bagi *consumer* dalam satu kali konsumsi.
3. Pengaturan pada *Market*, yaitu batas ukuran maksimum dan minimum *market*.
4. Pengaturan lain, yaitu jenis barang dan kecepatan proses simulasi.
5. Menu acak *Input* berfungsi untuk mengacak *input* pada halaman *Input*
6. Menu *About* Berfungsi untuk Menampilkan Halaman *About*
7. Menu Simulasi Berfungsi Untuk Melihat Proses Simulasi
8. Menu Keluar Berfungsi Untuk Keluar Dari Simulasi *Producer Consumer Problem*

Setelah pengaturan pada *form input*, proses simulasi dapat dimulai dengan menekan tombol ’Simulasi’. Selanjutnya, *form* simulasi akan muncul. Ketika proses simulasi sedang berjalan, *user* dapat menghentikan untuk sementara proses simulasi dengan menekan tombol ’Hentikan’. *User* juga dapat melihat laporan proses yang terjadi dalam simulasi dengan menekan tombol ’Laporan’. Laporan juga disediakan dalam bentuk tabel.

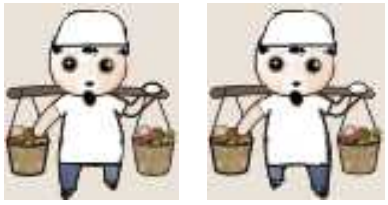


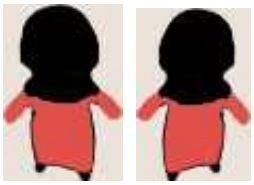
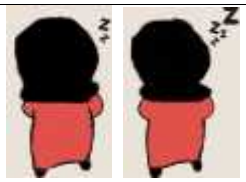
Gambar 4.4 *Diagram Alur kerja Simulasi Producer-Consumer Problem*

4.2.2 Penggambaran Objek Simulasi

Objek simulasi atau variabel yang terdapat dalam simulasi *Producer-Consumer Problem* diillustrasikan dalam bentuk objek gambar. Variabel utama dalam simulasi ini adalah *producer*, *consumer* dan *market*. Penggambaran variabel dilakukan dengan menggunakan aplikasi *Adobe Photoshop CS3*. Variabel beserta illustrasi objek gambarnya dapat dilihat pada tabel berikut ini:

Tabel 4.1 Tabel Gambar Objek Simulasi

Gambar	Keterangan
	Gambar *producer* berjalan ke bawah (tampak depan).

Gambar	Keterangan
	Gambar *producer* berjalan ke atas (tampak belakang).
	Gambar *producer* sedang tidur (*sleep*).
	Gambar *consumer* berjalan ke bawah (tampak depan).
	Gambar *consumer* berjalan ke atas (tampak belakang).
	Gambar *consumer* sedang tidur (*sleep*).
	Tempat *Producer* dan *consumer* menyimpan dan mengambil barang.

Gambar	Keterangan
	Tempat *producer* memproduksi barang
	Tempat *consumer* mengkonsumsi barang

4.2.3 Proses Simulasi Producer-Consumer Problem

Proses simulasi *Producer-Consumer Problem* adalah sebagai berikut:

1. *Producer* aktif memproduksi dan meletakkan *barang* ke *market* (*buffer*). Aksi ini akan menambah jumlah *barang* di dalam *market*.
2. *Consumer* aktif mengambil *barang* dari *market* (*buffer*) dan mengonsumsi *barang*. Aksi ini akan mengurangi jumlah *barang* di dalam *market*.
3. Apabila *market* telah penuh atau jumlah *barang* di dalam *market* telah mencapai batas maksimum, maka *producer* akan tidur (memanggil aksi *sleep*).
4. Apabila *consumer* mengambil *barang* dari *market* dan *producer* dalam keadaan *sleep*, maka *consumer* akan membangunkan (*wake-up*) *producer*.
5. Apabila *market* kosong atau jumlah *barang* di dalam *market* telah mencapai batas minimum, maka *consumer* akan tidur (memanggil aksi *sleep*).
6. Apabila *producer* meletakkan *barang* ke *market* dan *consumer* dalam keadaan *sleep*, maka *producer* akan membangunkan (*wake-up*) *consumer*.

Keenam poin di atas merupakan inti dari simulasi *Producer-Consumer Problem*. Pencegahan kondisi *deadlock* (*producer* ingin meletakkan *barang* ke *market* sedangkan *market* telah penuh atau *consumer* ingin mengambil *barang* dari *market* sedangkan *market* telah kosong) dihindari dengan metode *sleep* dan *wake up*. Masing-masing variabel akan memanggil aksi *sleep* untuk menghindari

kondisi *deadlock* dan akan dibangunkan variabel lainnya, ketika keadaan sudah tidak menyebabkan *deadlock*.

Proses simulasi yang dirancang dalam perangkat lunak menggunakan komponen *timer* yang terdapat dalam bahasa pemograman *Microsoft Visual Basic 6.0*. Fungsi dari komponen *timer* adalah untuk memeriksa keadaan suatu objek setiap satu interval waktu yang diatur kepadanya. *Timer* akan memeriksa dan memajukan keadaan objek ke keadaan berikutnya, sehingga terbentuk proses animasi. Misalkan, keadaan sekarang adalah *producer* berjalan ke bawah dengan kaki kiri, maka pada keadaan berikutnya, *timer* akan mengubah gambar tersebut dengan gambar *producer* berjalan ke bawah dengan kaki kanan. Demikian seterusnya.

4.2.4 Pengaturan Pergerakan Producer dan Consumer

4.2.4.1 Pengaturan Pergerakan *Producer*

Dalam perangkat lunak ini, pergerakan *producer* antara lain adalah:

1. Berada di dalam rumah.

   Producer diasumsikan sedang memproduksi *barang*. Gambar *producer* tidak terlihat pada daerah simulasi.

2. Berjalan menuju *market*.

   Producer diasumsikan telah siap memproduksi *barang* dan membawa *barang* hasil produksi ke *market*.

3. Meletakkan *barang* hasil produksi ke *market*

   Producer telah tiba di *market* dan meletakkan semua *barang* hasil produksinya ke *market*. Periksa apakah ukuran *market* dapat menampung semua *barang* hasil produksi *producer*. Jika ya, maka letakkan semua *barang* hasil produksi ke *market*. Jika tidak, maka letakkan sebagian *barang* hasil produksi ke *market* hingga *market* penuh (mencapai ukuran maksimum) dan panggil aksi tidur (*sleep*) untuk semua *producer*. *Update* jumlah *barang* pada *market* dan *producer*.

   Bangunkan (panggil aksi *wake-up*) untuk semua *consumer* (karena sudah terdapat *barang* di dalam *market*).

4. Berjalan kembali ke rumah.

   Producer telah selesai meletakkan semua *barang* hasil produksinya ke *market* dan berjalan kembali ke rumah.

4.2.4.2 Pengaturan Pergerakan *Consumer*

Dalam perangkat lunak ini, pergerakan *consumer* antara lain adalah:

1. Berada di dalam rumah.

   Consumer diasumsikan sedang mengonsumsi *barang* hasil pembelian dari *market*. Gambar *consumer* tidak terlihat pada daerah simulasi.

2. Berjalan menuju *market*.

   Consumer diasumsikan telah habis mengonsumsi *barang* dan mulai berjalan ke *market* untuk membeli *barang*.

3. Membeli *barang* dari *market*

   Consumer telah tiba di *market* dan membeli sejumlah *barang* dari *market*. Periksa apakah ukuran *market* dapat memenuhi permintaan *barang consumer*. Jika ya, maka ambil semua *barang* dari *market*. Jika tidak, maka ambil sebagian *barang* dari *market* hingga *market* mencapai ukuran minimum dan panggil aksi tidur (*sleep*) untuk semua *consumer*. *Update* jumlah *barang* pada *market* dan *consumer*.Bangunkan (panggil aksi *wake-up*) untuk semua *producer* (karena beberapa *barang* sudah diambil dari dalam *market*).

4. Berjalan kembali ke rumah.

   Consumer telah selesai membeli dan mengambil *barang* dari *market* dan berjalan kembali ke rumah.

4.2.5 Tampilan

Terdapat enam buah *form* pada simulasi ini yaitu:

4.2.5.1 Form Splash Screen

Form splash screen berisi nama atau judul perangkat lunak dan nama pembuat perangkat lunak. *Form* ini tampil pertama kali ketika perangkat lunak dijalankan. *Form* ini dimaksudkan untuk memperkenalkan perangkat lunak secara singkat ketika dijalankan *user*.

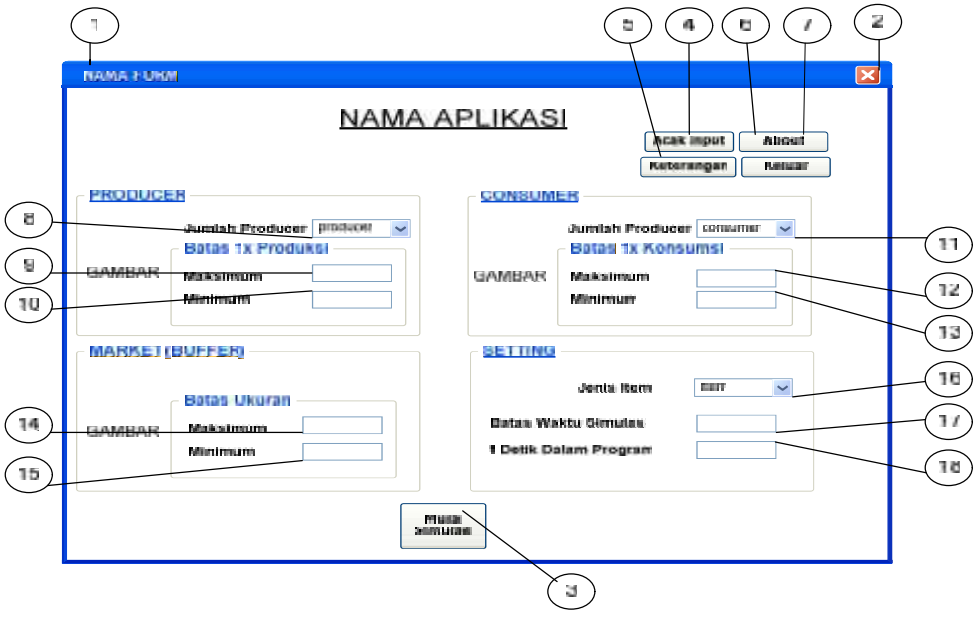
Gambar 4.5 Rancangan *Form Splash Screen*

Keterangan:

Nama perangkat lunak.

4.2.5.2 Form Input

Form input berfungsi untuk memasukkan *input* perangkat lunak. Input perangkat lunak berupa: jumlah *producer*, jumlah *customer*, batas maksimum dan minimum bagi *producer* dalam satu kali produksi, batas maksimum dan minimum bagi *consumer* dalam satu kali konsumsi, batas ukuran maksimum dan minimum *market*, banyak jenis barang dan kecepatan proses simulasi.



Gambar 4.6 Rancangan *Form Input*

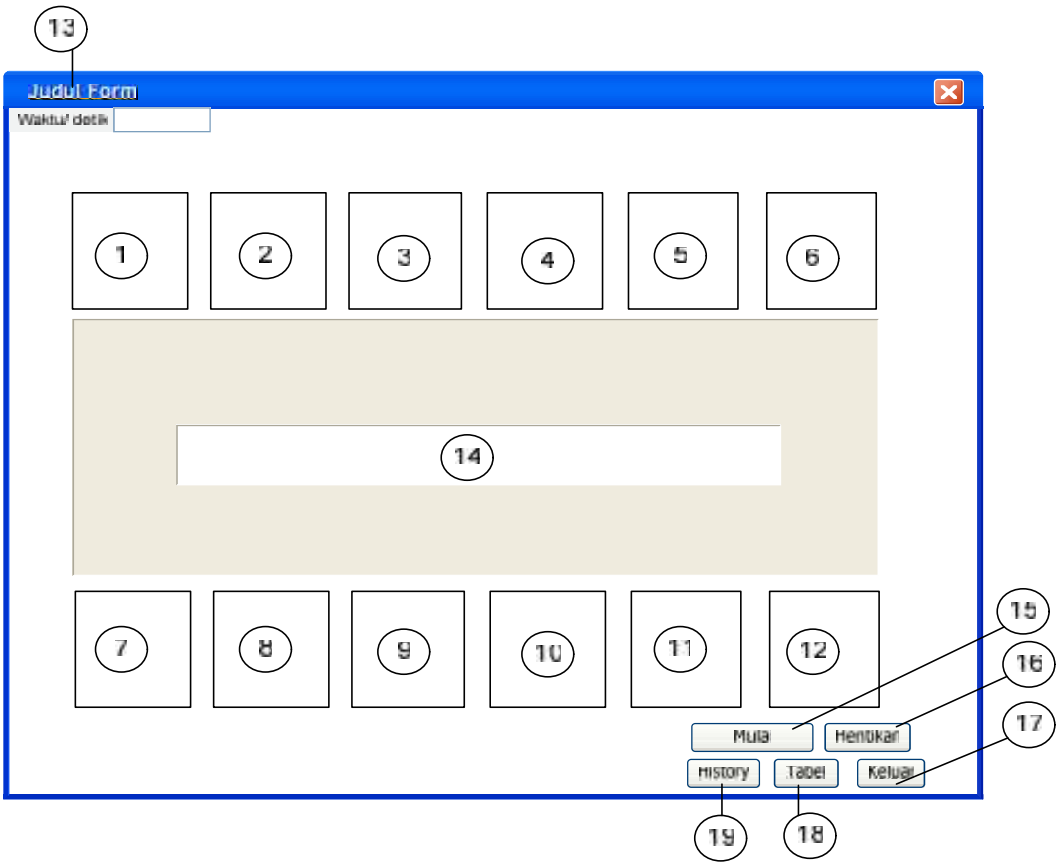
Keterangan :

Tabel 4.2 Tabel Rancangan Form Input

NO	Nama Objek	Keterangan
1	*title bar*.	
2	Exit	untuk menutup *form*
3	Simulasi	untuk menampilkan *form* simulasi
4	Acak Input	untuk mengacak semua nilai *input*
5	Keterangan	Untuk Menampilkan Keterangan dari Simulasi
6	About	untuk menampilkan *form* About
7	Keluar	untuk menutup perangkat lunak
8	*Combobox*	untuk memilih jumlah *producer*
9	*Textbox*	untuk memasukkan batas maksimum satu kali produksi *barang* oleh *producer*
10	*Textbox*	untuk memasukkan batas minimum satu kali produksi *barang* oleh *producer*
11	*combobox*	untuk memilih jumlah *customer*
12	*Textbox*	untuk memasukkan batas maksimum satu kali konsumsi *barang* oleh *consumer*.
13	*Textbox*	untuk memasukkan batas minimum satu kali konsumsi *barang* oleh *consumer*.
14	*Textbox*	untuk memasukkan batas ukuran maksimum *market*.
15	*Textbox*	untuk memasukkan batas ukuran minimum *market*.
16	*combobox*	untuk memilih banyak jenis *barang*.
17	*Textbox*	untuk memasukkan batas waktu simulasi.
18	*Textbox*	untuk memasukkan kecepatan proses.

4.2.5.3 Form Simulasi

Form Simulasi berfungsi untuk menampilkan proses simulasi *Producer-Consumer Problem*. Dalam *form* ini, terdapat enam buah pintu keluar *producer*, 6 buah pintu keluar *consumer* dan sebuah *market* (illustrasi dari *buffer*), sebagai tempat untuk meletakkan dan mengambil *barang*.



Gambar 4.7 Rancangan *Form* Simulasi

Keterangan:

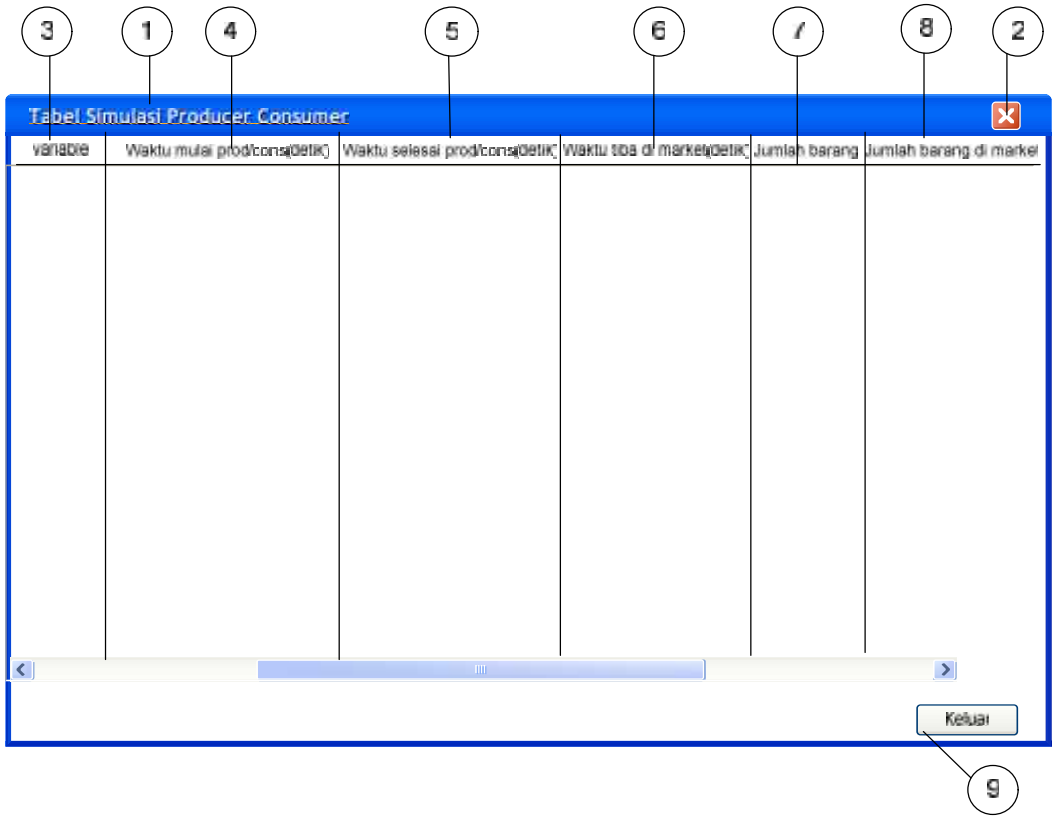
Tabel 4.3 Tabel Rancangan Form Simulasi

NO	Nama Objek	Keterangan
1	pintu keluar *producer*-1	
2	pintu keluar *producer*-2	
3	pintu keluar *producer*-3	
4	pintu keluar *producer*-4	
5	pintu keluar *producer*-5	
6	pintu keluar *producer*-6	
7	pintu keluar *customer*-1	
8	pintu keluar *customer*-2	
9	pintu keluar *customer*-3	
10	pintu keluar *customer*-4	
11	pintu keluar *customer*-5	
12	pintu keluar *customer*-6	
13	Label	menampilkan waktu simulasi.
14	Label	menampilkan jumlah *barang* dalam *market*.
15	tombol ‘Mulai’	untuk menjalankan proses

		simulasi.
16	tombol ‘Hentikan’	untuk menghentikan sementara proses simulasi.
17	tombol ‘Keluar’	untuk menutup *form* simulasi dan kembali ke *form input*.
18	tombol ‘Tabel’	untuk menampilkan *form* tabel laporan.
19	tombol ‘Laporan’	untuk menampilkan laporan proses yang terjadi dengan menampilkan form Laporan.

4.2.5.4 Form Tabel Simulasi

Form Tabel Simulasi berfungsi untuk menampilkan tabel laporan yang mencatat setiap waktu mulai produksi/konsumsi, waktu selesai produksi/ konsumsi, waktu tiba di *market* dan penambahan serta pengurangan barang dalam *market* selama proses simulasi berlangsung.



Gambar 4.8 Rancangan *Form* Tabel Simulasi

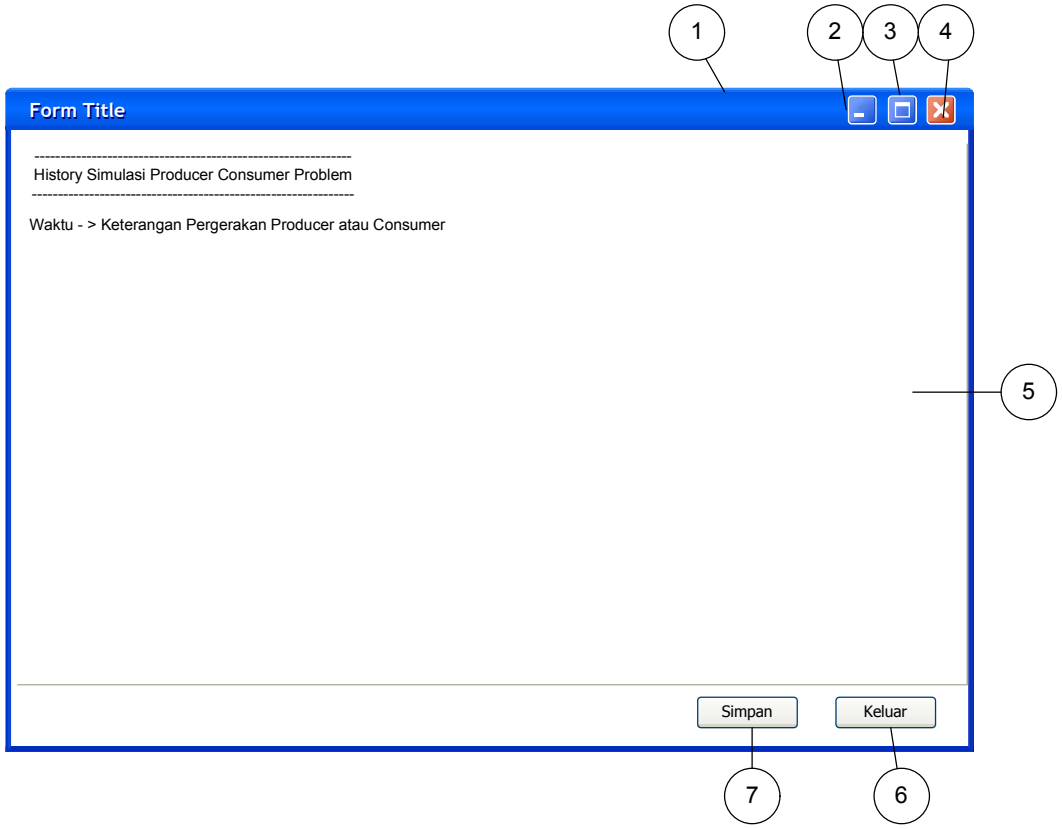
Keterangan:

Tabel 4.4 Tabel Rancangan Form Tabel Simulasi

NO	Nama Objek	Keterangan
1	*title bar*	
2	tombol 'Exit'	untuk menutup *form*.
3	"Variable"	nama *producer* dan *consumer*
4	"waktu mulai prod/kons (detik)"	waktu *producer* dan *consumer* mulai memproduksi atau mengkonsumsi barang
5	"waktu selesai prod/kons (detik"	waktu *producer* dan *consumer* selesai memproduksi atau mengkonsumsi barang
6	"waktu tiba di *market* (detik)"	Waktu *producer* dan *consumer* tiba di *market*
7	"jumlah barang"	jumlah barang yang di simpan atau di konsumsi oleh *producer* dan *consumer*
8	"Jumlah barang di *market*"	jumlah barang yang terdapat pada *market*
9	tombol 'Keluar'	untuk menutup *form*.

4.2.5.5 Form History

Form History berfungsi untuk mencatat dan menampilkan laporan proses yang telah terjadi pada proses simulasi. Laporan proses berupa waktu transaksi dan jenis transaksi yang terjadi (peletakkan *barang* ke *market*, pengambilan *barang* ke *market*, aksi *sleep* dan *wake-up* yang dipanggil oleh *producer* ataupun *consumer*).



Gambar 4.9 Rancangan *Form* History

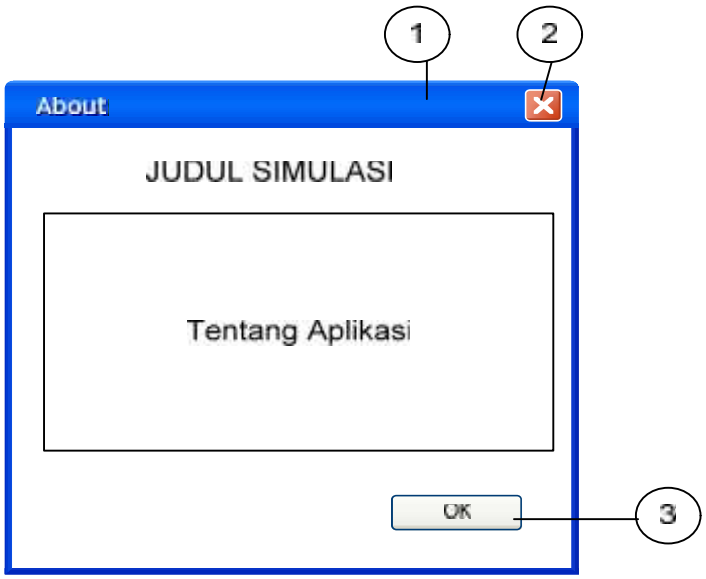
Keterangan:

Tabel 4.5 Tabel Rancangan Form History

NO	Nama Objek	Keterangan
1	*title bar*	
2	tombol 'Minimize'	untuk me-*minimize form*.
3	tombol 'Maximize'	untuk me-*maximize form*.
4	tombol 'Exit'	untuk menutup *form*.
5	*Textbox*	untuk menampilkan laporan proses.
6	tombol 'Simpan'	untuk menyimpan laporan proses dalam format teks.
7	tombol 'Keluar'	untuk menutup *form*.

4.2.1.6 Form About

Form About berfungsi untuk menampilkan judul/nama perangkat lunak, dan keterangan tentang Simulasi *Producer-Consumer Problem*.



Gambar 4.10 Rancangan *Form About*

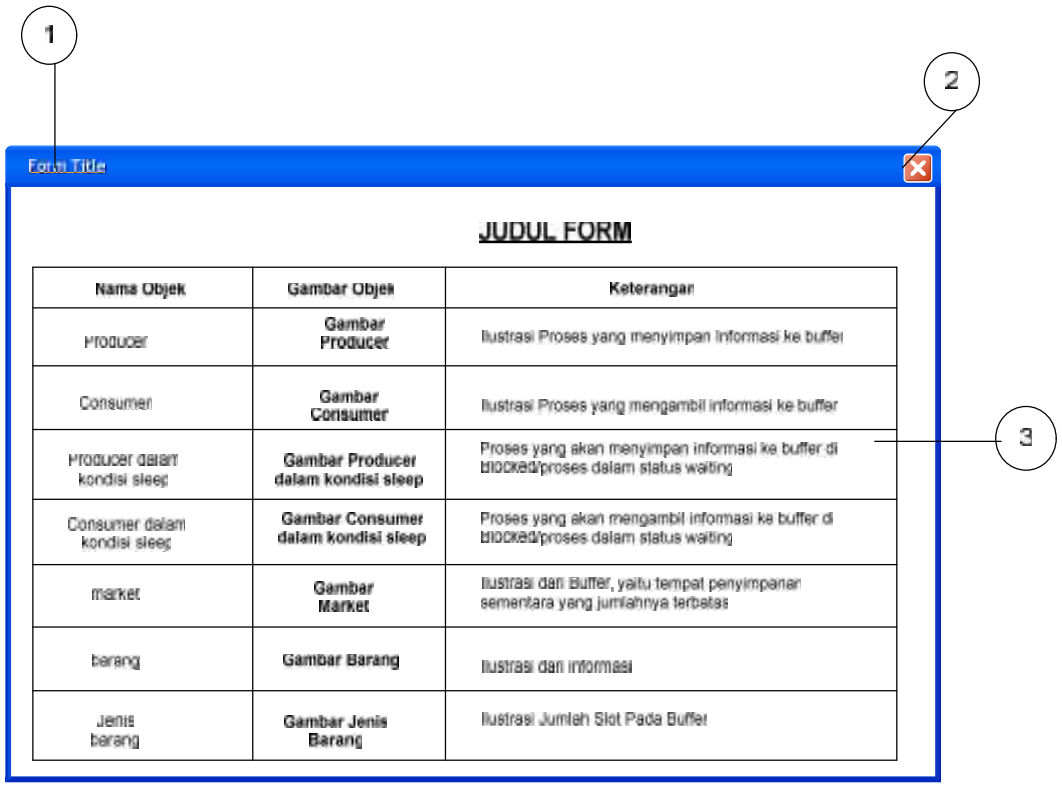
Keterangan:

Tabel 4.7 Tabel Rancangan Form About

NO	Nama Objek	Keterangan
1	*title bar*.	
2	tombol 'Exit',	untuk menutup *form*.
3	tombol 'OK',	untuk menutup *form About*.

4.2.5.7 Form Keterangan

Form keterangan berfungsi untuk menampilkan keterangan dari masing-masing objek yang ada pada simulasi *producer-consumer problem.*



Gambar 4.11 Rancangan *Form* Simulasi

Keterangan:

Tabel 4.8 Tabel Rancangan Form Simulasi

NO	Nama Objek	Keterangan
1	*title bar*.	
2	tombol ‘Exit’,	untuk menutup *form*.
3	Keterangan	Keterangan dari Simulasi

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sistem operasi secara umum adalah pengelola seluruh sumber daya yang terdapat pada sistem komputer dan menyediakan sekumpulan layanan ke pemakai sehingga memudahkan penggunaan serta pemanfaatan sumber daya pada sistem komputer. Sistem operasi bertugas untuk mengatur proses-proses yang berjalan dalam jangka waktu yang sama tanpa boleh saling bertabrakan satu dengan yang lainnya. Proses-proses yang berinteraksi memerlukan *sinkronisasi* agar terhindar dari kondisi *deadlock*.

Deadlock dapat didefinisikan sebagai pemblokiran permanen sejumlah proses yang berkompetisi dalam mendapatkan sumberdaya sistem atau yang berkomunikasi satu dengan yang lainnya. *Deadlock* disebabkan karena proses yang satu menunggu sumber daya yang sedang dipegang oleh proses lain. Dengan kata lain setiap proses dalam set menunggu untuk sumber yang hanya dapat dikerjakan oleh proses lain. *Deadlock* terjadi ketika proses-proses mengakses secara ekslusif sumber daya. Semua *deadlock* yang terjadi melibatkan persaingan memperoleh sumber daya ekslusif oleh dua proses atau lebih.

Sinkronisasi adalah proses pengaturan jalannya beberapa proses pada saat yang bersamaan. Tujuan utama *sinkronisasi* adalah untuk mengatur urutan jalannya proses-proses yang berbeda sehingga dapat berjalan dengan lancar. Salah satu ilustrasi dari masalah sinkronisasi adalah kasus *producer-consumer problem*. Diibaratkan *producer* dan *consumer* (ilustrasi dari proses) berbagi sebuah market (ilustrasi dari *buffer*) dengan kapasitas yang terbatas. *Produser* menyimpan barang ke market, Sedangkan *consumer* mengambil barang dari market. Masalah bagi *produser* ketika ingin menaruh barang yang baru tetapi market sudah penuh, sedangkan masalah bagi *consumer* terjadi ketika *consumer* ingin mengambil barang sementara *market* kosong. Kejadian ini mengakibatkan *deadlock* sebab kedua proses (*producer dan consumer*) akan saling menunggu kejadian yang tidak

akan pernah terjadi. Kedua proses di atas memerlukan sinkronisasi agar terhindar dari masalah *deadlock.*

Tetapi proses-proses yang terjadi pada sistem operasi bersifat abstrak dan relatif sulit dipahami. Maka dari itu di butuhkan suatu pendekatan untuk mengilustrasikan dan menvisualisasikan proses-proses yang terjadi pada sistem operasi sehingga dapat membantu dalam memahami masalah *producer-consumer problem*. Pada penelitian ini di rancang suatu perangkat lunak yang dapat mengilustrasikan dan menvisualisasikan kasus *producer-consumer problem* dalam bentuk simulasi.

Simulasi adalah suatu teknik meniru operasi atau proses yang terjadi dalam suatu sistem dengan bantuan perangkat komputer dan dilandasi oleh beberapa asumsi tertentu sehingga sistem tersebut bisa dipelajari secara ilmiah. Kasus *producer-consumer problem* di simulasikan sebagai sebuah pasar, dimana terdapat *producer, consumer* dan market. *Producer* adalah variable yang memasukkan barang ke market, sementara *consumer* adalah variable yang membeli barang dari market.

Pada simulasi *producer-consumer problem* ini, penulis menggunakan metode *sleep and wake up* untuk menghindari kondisi *deadlock*. Mekanismenya proses akan di *blok*/tidur (sleep) apabila tidak bisa memasuki *buffer* dan akan dibangunkan *(wake up)*/*ready* apabila *buffer* yang diperlukan telah tersedia.

Dari penjelasan diatas, penulis bermaksud untuk membuat visualisasi dari masalah *producer-consumer problem* tersebut dengan merancang suatu perangkat lunak berupa Simulasi Pencegahan *Deadlock* Pada Kasus *Producer-Consumer Problem* Sebagai Ilustrasi Dalam Sistem Operasi.

1.2 Rumusan Masalah

Rumusan masalah pada Tugas Akhir ini adalah bagaimana merancang suatu perangkat lunak yang dapat memodelkan kasus *producer-consumer problem* dalam bentuk simulasi.

1.3 Batasan Masalah

Batasan Masalah pada Tugas Akhir ini adalah:

1. Simulasi akan menampilkan proses yang terjadi dalam bentuk animasi 2 dimensi.
2. Jumlah *producer* dibatasi minimal 1 orang dan maksimal 6 orang dan Jumlah *consumer* dibatasi minimal 1 orang dan maksimal 6 orang.
3. Batas maksimum dan minimum *Producer* dalam 1 kali produksi minimum 1 Barang dan maksimum 99 Barang.
4. Batas maksimum dan minimum *Consumer* dalam 1 kali konsumsi minimum 1 Barang dan maksimum 99 Barang.

1.4 Tujuan

Tujuan dari penulisan Tugas Akhir ini adalah:

1. Merancang dan mengimplementasikan suatu perangkat lunak yang dapat mensimulasikan kasus *producer-consumer problem* dengan menggunakan Metode *Sleep and wakeup*.
2. Aplikasi ini dapat membantu pemahaman terhadap kasus kasus *producer-consumer problem* serta dapat menjadi salah satu alternatif dalam mendukung proses pembelajaran, terutama mengenai topik *sinkronisasi*.

1.5 Sistematika Penulisan

Sistematika penulisan dalam Tugas akhir ini adalah sebagai berikut:

Bab–bab tersebut diantaranya adalah:

BAB I PENDAHULUAN

Merupakan deskripsi umum dari tugas akhir ini, yang meliputi: latar belakang masalah, rumusan masalah, batasan masalah, tujuan penyusunan tugas akhir, metodologi penelitian serta sistematika pembahasan tugas akhir.

BAB II LANDASAN TEORI

Dalam Bab ini berisi teori atau gambaran umum serta kebutuhan sistem yang berkaitan/ berhubungan dengan Aplikasi Simulasi *Producer-Consumer*

Problem. Teori-teori tersebut, antara lain Teori mengenai sistem operasi, komponen sistem operasi, proses, *sinkronisasi proses,* metode *Sleep and wake up, deadlock*, model, simulasi dan perancangan sistem.

BAB III METODOLOGI PENELITIAN

Berisi tentang langkah-langkah dalam melaksanakan Tugas Akhir yang dikerjakan seperti studi pustaka, perumusan masalah, analisa, perancangan, implementasi, pengujian serta kesimpulan dan saran.

BAB IV ANALISA DAN PERANCANGAN

Bab ini berisi tentang pembahasan analisa sistem yaitu analisa Permasalahan *kasus producer-consumer problem,* analisa simulasi, analisa metode, perancangan alur kerja simulasi dan perancangan *interface.*

BAB V IMPLEMENTASI DAN PENGUJIAN

Berisi tentang *interface* sistem serta analisis dan pengujian sistem yang telah dibuat. Pada tahap ini penggujian dilakukan dengan menggunakan metode pengujian *black box.*

BAB VI PENUTUP

Dalam bab ini berisi beberapa kesimpulan yang diperoleh dari pembahasan tentang simulasi pencegahan *deadlock* pada kasus *producer-consumer problem* sebagai ilustrasi dalam sistem operasi, serta saran untuk pengembangan selanjutnya.

BAB II
LANDASAN TEORI

2.1 Sistem Operasi

Sistem operasi merupakan suatu perangkat lunak yang mengelola seluruh sumber daya sistem komputer dan penyedia layanan pada *user*, yang sekaligus bertindak sebagai *interface* antara *user* dan sistem komputer. Sistem operasi merupakan sebuah penghubung antara pengguna dari komputer dengan perangkat keras komputer. Sebelum ada sistem operasi, orang hanya mengunakan komputer dengan menggunakan sinyal analog dan sinyal digital. Seiring dengan berkembangnya pengetahuan dan teknologi, pada saat ini terdapat berbagai sistem operasi dengan keunggulan masing-masing. Untuk lebih memahami sistem operasi maka sebaiknya perlu diketahui terlebih dahulu beberapa konsep dasar mengenai sistem operasi itu sendiri.

2.1.1 Tugas Utama Sistem Operasi

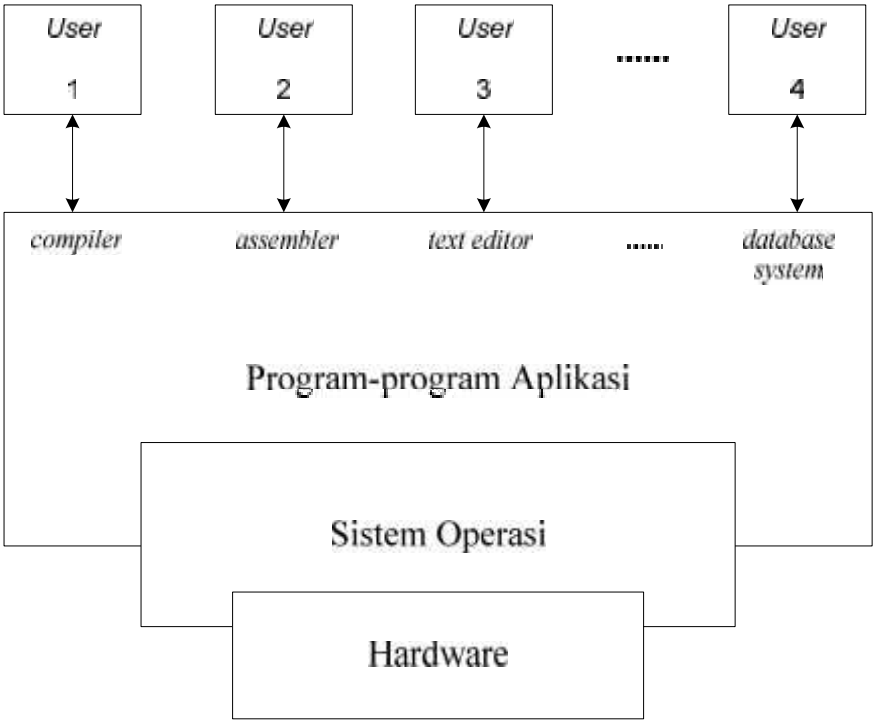
Sistem operasi mempunyai tiga tujuan dasar:

1. Efisiensi, yaitu sistem operasi memungkinkan sumber daya sistem komputer untuk di gunakan dengan cara yang efisien
2. Kemudahan, yaitu sistem operasi membuat komputer mudah di gunakan
3. Kemampuan berevolusi, maksudnya adalah sistem operasi harus disusun sedemikian rupa sehingga memungkinkan pengembangan yang efektif, pengujian dan penerapan fungsi-fungsi sistem yang baru tanpa menggaggu layanan yang telah ada (Pangera dan Arriyus, 2008).

2.1.2 Komponen Sistem operasi

Sistem operasi adalah bagian yang sangat penting bagi semua sistem komputer. Secara umum, sistem komputer terbagi atas *hardware*, sistem operasi, program aplikasi, dan *user*. *Hardware* terdiri atas CPU, memori dan I/O *device* yang merupakan *resources* dasar. Program aplikasi berisi *compiler*, basis data, *games* dan program-program bisnis, yang merupakan alat dimana *resource* akan diakses untuk menyelesaikan masalah *user* (Kusnadi dan Kuswoyo,2008).

Komponen-komponen sistem komputer tersebut ditunjukkan oleh gambar 2.1 berikut ini:



Gambar 2.1 Abstraksi komponen-komponen Komputer

2.1.3 Sasaran Sistem Operasi

Sistem operasi mempunyai tiga sasaran utama yaitu *kenyamanan* membuat penggunaan komputer menjadi lebih nyaman, *efisien* penggunaan sumber-daya sistem komputer secara efisien, serta mampu *berevolusi* sistem operasi harus dibangun sehingga memungkinkan dan memudahkan pengembangan, pengujian serta pengajuan sistem-sistem yang baru (Stalling,2003).

2.2 Komponen Sistem Operasi

Menurut Avi Silberschatz, Peter Galvin, dan Greg Gagne, pada umumnya sebuah sistem operasi modern mempunyai komponen sebagai berikut:

1. Manajemen Proses.

Proses adalah program yang sedang dieksekusi. Suatu proses membutuhkan satu atau beberapa sumber daya untuk menyelesaikan tugasnya.

Sumber daya tersebut dapat berupa *CPU time*, memori, berkas-berkas, atau perangkat-perangkat I/O. Sistem operasi bertanggung jawab atas aktivitas-aktivitas yang berkaitan dengan manajemen proses seperti:

a. Pembuatan dan penghapusan proses pengguna dan sistem proses.
b. Menunda atau melanjutkan proses.
c. Menyediakan mekanisme untuk proses sinkronisasi.
d. Menyediakan mekanisme untuk proses komunikasi.
e. Menyediakan mekanisme untuk penanganan *deadlock*.

2. Manajemen Memori Utama.

Memori utama merupakan istilah generik yang merujuk pada media penyimpanan data sementara pada komputer. Setiap program dan data yang sedang diproses oleh prosesor akan disimpan di dalam memori utama. Data yang disimpan dalam memori bersifat sementara, karena data yang disimpan di dalamnya akan tersimpan selama komputer tersebut masih dialiri daya (dengan kata lain, komputer itu masih hidup). Ketika komputer itu direset atau dimatikan, data yang disimpan dalam memori utama akan hilang. Oleh karena itulah, sebelum mematikan komputer, semua data yang belum disimpan ke dalam media penyimpanan permanen (umumnya bersifat media penyimpanan permanen berbasis disk, semacam *hard disk* atau *floppy disk*), sehingga data tersebut dapat dibuka kembali pada lain waktu (Shelly,2010).

Memori utama umumnya diimplementasikan dalam bentuk *Random Access Memory* (RAM), yang bersifat dinamis (DRAM). Mengapa disebut *Random Access*, adalah karena akses terhadap lokasi-lokasi di dalamnya dapat dilakukan secara acak (random), bukan secara berurutan (sekuensial). Meskipun demikian, kata *random access* dalam RAM ini sering menjadi salah kaprah. Sebagai contoh, memori yang hanya dapat dibaca (ROM), juga dapat diakses secara random, tetapi ia dibedakan dengan RAM karena ROM dapat menyimpan data tanpa kebutuhan daya dan tidak dapat ditulisi sewaktu-waktu. Selain itu, hard disk yang juga merupakan salah satu media penyimpanan juga dapat diakses secara random, tapi ia tidak digolongkan ke dalam *Random Access Memory*.

Penggunaan Memory utama dalam sistem komputer adalah *Arithmetic Logic Unit* (ALU), *Control Circuitry, Storage Space* dan piranti Input/Output. Jika tanpa *memory*, maka komputer hanya berfungsi sebagai digital signal processing devices, contohnya kalkulator atau media player. Kemampuan *memory* untuk menyimpan data, instruksi dan informasi lah yang membuat komputer dapat disebut sebagai *general-purpose* komputer. Komputer merupakan piranti digital, maka informasi disajikan dengan sistem bilangan *binary.* Teks, angka, gambar, audio dan video dikonversikan menjadi sekumpulan bilangan *binary (binary* digit atau disingkat bit). Sekumpulan bilangan binary dikenal dengan istilah BYTE, dimana 1 byte = 8 bits. Semakin besar ukuran memory-nya maka semakin banyak pula informasi yang dapat disimpan di dalam komputer (storage devices). Berikut ini beberapa gambar yang bisa mewakili bagaimana cara informasi disimpan dalam memory dan bagaimana data ditransfer dari satu bagian ke bagian lainnya.

Sistem operasi bertanggung jawab atas aktivitas-aktivitas yang berkaitan dengan manajemen memori seperti:

a. Menjaga *track* dari memori yang sedang digunakan dan siapa yang menggunakannya.
b. Memilih program yang akan di-*load* ke memori.
c. Mengalokasikan dan meng-dealokasikan ruang memori sesuai kebutuhan.

3. Manajemen *Secondary Storage*.

Data yang disimpan dalam memori utama bersifat sementara dan ukurannya sangat kecil jika dibandingkan dengan keseluruhan data yang terdapat dalam komputer. Oleh karena itu, untuk meyimpan keseluruhan data dan program komputer dibutuhkan *secondary storage* yang bersifat non volatil dan mampu menampung banyak data. Contoh dari *secondary storage* adalah *harddisk*, disket, dan lain-lain. Sistem operasi bertanggung jawab atas aktivitas-aktivitas yang berkaitan dengan *disk-management* seperti *free-space management*, alokasi penyimpanan, penjadwalan disk.

4. Manajemen Sistem I/O.

Fungsi ini sering disebut *device manager*, dimana sistem operasi menyediakan "*device driver*" yang umum sehingga operasi I/O dapat seragam

(membaca atau menuliskan data tanpa mempedulikan mekanisme kerja yang berbeda dari perangkat-perangkat I/O yang ada). Contoh: pengguna menggunakan operasi yang sama untuk membaca berkas pada *hard-disk*, CD-ROM dan *floppy disk*. Komponen untuk sistem I/O, yaitu:

a. *Buffer*: penampungan sementara data dari/ ke perangkat I/O.
b. *Spooling*: melakukan penjadualan pemakaian I/O sistem supaya pemakaian I/O lebih efisien (antrian dan sebagainya).
c. *Driver*: penerjemah instruksi antara sistem operasi dan I/O untuk dapat melakukan operasi tertentu. Setiap perangkat keras I/O memiliki *driver* yang berbeda-beda.

5. Manajemen *File*.

File adalah kumpulan informasi yang dibuat dengan tujuan tertentu. *File* disimpan dalam struktur yang bersifat hirarkis, seperti direktori. Dalam manajemen *file*, sistem operasi bertanggung jawab dalam melakukan:

a. Pembuatan dan penghapusan berkas.
b. Pembuatan dan penghapusan direktori.
c. Pemanipulasian berkas dan direktori.
d. Pemetaan berkas ke *secondary storage*.
e. *Backup* berkas ke media penyimpanan yang permanen (*non-volatile*).

6. Sistem Proteksi.

Proteksi mengacu pada mekanisme untuk mengontrol akses yang dilakukan oleh program, prosesor, atau pengguna ke sistem sumber daya. Mekanisme proteksi harus:

a. Membedakan antara penggunaan yang sudah diberi izin dan yang belum.
b. Menetapkan pembatasan atau pengaturan yang telah ditentukan (*specify the controls to be imposed*).
c. Menyediakan tata cara pelaksanaan (*provide a means of enforcement*).

7. Sistem Jaringan.

Sistem ini untuk mendukung penggunaan jaringan. Sistem ini umumnya kini telah terpadu dalam sistem operasi karena kebutuhan kinerjanya serta kebutuhan komputasi telah menghendaki kemampuan ini terdapat di dalam sistem

komputer. Sistem ini menyediakan akses pengguna ke bermacam sumber-daya sistem. Sistem ini membawa keuntungan dalam hal:

a. Kecepatan komputer yang meningkat (*computation speed-up*).
b. Ketersediaan data meningkat (*increased data availability*).
c. Realibilitas dapat ditingkatkan (*enhanced reliability*).

8. *User Interface (Shell)*.

Sistem Operasi menunggu instruksi dari pengguna (*command driven*). Program yang membaca instruksi dan mengartikan *control statements* disebut *control-card interpreter* atau *command-line interpreter*. *User Interface* sangat bervariasi dari satu sistem operasi ke sistem operasi yang lain dan disesuaikan dengan tujuan dan teknologi *I/O devices* yang ada. Contohnya: *Command Line Interface* (CLI), *Graphical User Interface* (GUI), dan lain-lain.

2.3 Proses

Proses merupakan konsep pokok di sistem operasi. Konsep ini pertama kali dipakai di sistem operasi Multics pada tahun 60-an. Tema utama perancangan sistem operasi semuanya berkaitan dengan manajemen proses.

Secara informal, proses adalah program dalam eksekusi. Proses merupakan unit kerja terkecil yang secara individu memiliki sumber daya-sumber daya dan dijadwalkan sistem operasi. Sistem operasi mengelola semua proses dalam komputer dan mengalokasikan sumber daya ke proses-proses sesuai kebijaksanaan yang digunakan/diterapkan untuk memenuhi sasaran sistem (Pangera dan Ariyus, 2008).

Program sendiri bukanlah suatu proses. Suatu program adalah satu entitas pasif; seperti isi dari sebuah berkas yang disimpan dalam disket. Suatu proses merupakan suatu entitas aktif, dengan sebuah *program counter* yang menunjuk pada instruksi selanjutnya untuk dijalankan dan seperangkat sumber daya/*resource* yang berkenaan dengannya.

Walau lebih dari satu proses dapat dihubungkan dengan sebuah program yang sama, program tersebut dianggap merupakan urutan- urutan eksekusi yang berbeda. Sebagai contoh, beberapa pengguna dapat menjalankan *copy* yang

berbeda pada *mail program*, atau pengguna yang sama dapat meminta banyak *copy* dari *program editor*. Tiap-tiap proses ini adalah proses yang berbeda walaupun bagian tulisan *text* yang dikerjakan adalah sama. Adalah hal yang umum untuk memiliki proses yang menghasilkan banyak proses saat suatu proses tersebut dieksekusi.

Program adalah sederetan instruksi yang diberikan kepada suatu komputer. Sedangkan *proses* adalah suatu bagian dari program yang berada pada status tertentu dalam rangkaian eksekusinya. Pada Sistem Operasi modern, pada satu saat tidak seluruh program dimuat dalam memori, tetapi hanya satu bagian saja dari program tersebut. Sedangkan bagian lain dari program tersebut tetap beristirahat di media penyimpan disk. Hanya pada saat dibutuhkan saja, bagian dari program tersebut dimuat di memory dan dieksekusi oleh prosesor. Hal ini sangat menghemat pemakaian memori.

Beberapa sistem hanya menjalankan satu proses tunggal dalam satu waktu, sedangkan yang lainnya menjalankan multi-proses dalam satu waktu. Padahal sebagian besar sistem komputer hanya memiliki satu prosesor, dan sebuah prosesor hanya dapat menjalankan satu instruksi dalam satu waktu. Sesungguhnya pada granularity yang sangat kecil, prosesor hanya menjalankan satu proses dalam satu waktu, kemudian secara cepat ia berpindah menjalankan proses lainnya, dan seterusnya. Sehingga bagi penglihatan dan perasaan pengguna manusia, seakan-akan prosesor menjalankan beberapa proses secara bersamaan.

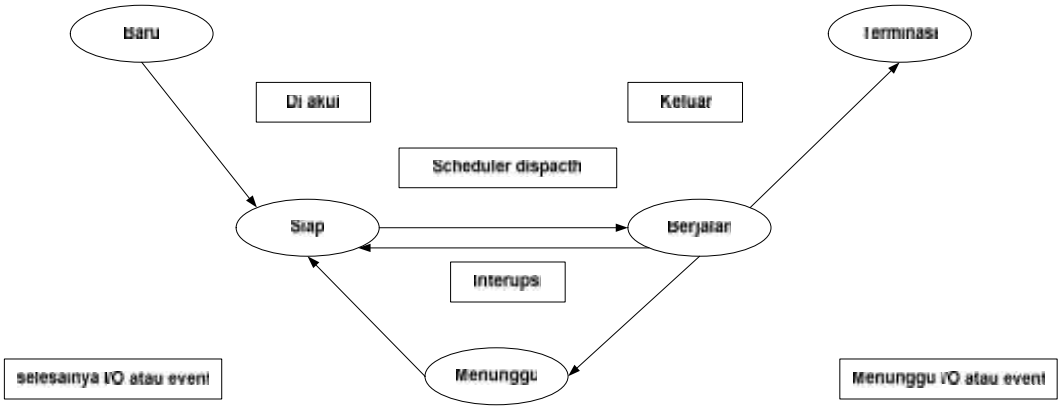
2.3.1 Keadaan Proses

Saat proses dieksekusi, proses tersebut dapat berubah statusnya (*state*). Status dari sebuah proses didefinisikan oleh aktivitas yang sedang bekerja dalam proses tersebut. Status dari suatu proses dapat berupa:

1. ***New*:** Proses baru diciptakan.
2. ***Running*:** Proses sedang dikerjakan.
3. ***Waiting*:** Proses sedang menunggu sejumlah kejadian untuk terjadi (seperti sebuah penyelesaian I/O atau penerimaan sebuah tanda/*signal*).
4. ***Ready*:** Proses sedang menunggu untuk dieksekusi pada sebuah prosesor.

5. ***Terminated*:** Proses telah selesai melaksanakan tugasnya.

Adalah penting untuk menyadari bahwa hanya satu proses dapat berjalan pada suatu prosesor pada satu saat. Namun, lebih dari satu proses yang dapat berada dalam status *ready* atau *waiting*. Diagram yang berkaitan dangan status-status tersebut dapat dilihat pada gambar 2.1.



Gambar 2.2 Diagram keadaan Proses

2.3.2 Process Control Block (PCB)

Parameter-parameter suatu proses dicatat dalam bagian yang disebut *process control block* (PCB). PCB berisikan informasi-informasi yang berhubungan dengan sebuah proses, yaitu:

1. Keadaan proses, yaitu keadaan pada proses: *new*, *ready*, *running*, *waiting*, *halted*, dan juga banyak lagi.
2. *Program counter*, mengindikasikan alamat (*address*) dari perintah selanjutnya untuk dijalankan untuk proses ini.
3. CPU register, yaitu *register-register* arsitektural yang digunakan oleh proses beserta status dan data di dalamnya.
4. Informasi manajemen memori, dapat berupa informasi nilai dari dasar dan batas *register*, tabel *page*, atau tabel segmen tergantung pada sistem memori yang digunakan oleh sistem operasi.
5. Informasi pencatatan, berupa informasi CPU *time* dan waktu riil yang digunakan, batas waktu, jumlah *job*, dan sebagainya.
6. Informasi status I/O, termasuk daftar dari perangkat I/O yang digunakan oleh proses ini, daftar *file* yang digunakan, dan sebagainya.

Gambaran suatu *process control block* (PCB) dapat dilihat pada gambar 2.3 berikut.

Gambar 2.3 *Process Control Block*

2.3.3 Threads

Thread adalah abstraksi dari unit aktivitas (penjadwalan). Sebuah proses dapat memiliki lebih dari satu *thread*. Sistem yang memungkinkan lebih dari satu *thread* dieksekusi secara bersamaan disebut *multithreading*. Pada *multiprocessors*, *thread-thread* dalam satu proses dapat berjalan secara parallel (Sri Kusuma Dewi,2000).

Thread (sering disebut *Light Weight Process* (LWP)) adalah unit dasar utilisasi pemroses dan berisi *program counter*, *register set* dan *stack space*. *Thread-thread* dalam satu proses berbagi (memakai bersama) bagian kode, data dan sumber daya sistem operasi seperti *file* dan *signal*. *Multithreading* merupakan upaya meningkatkan kinerja sistem komputer, disebabkan:

1. Penciptaan *thread* baru lebih cepat dibanding penciptaan proses baru.
2. Terminasi *thread* lebih cepat dibanding pengakhiran proses.
3. Pengalihan ke *thread* lain di satu proses lebih cepat dibanding beralih dari satu proses ke proses lain.
4. *Thread-thread* pada satu proses dapat berbagi kode, data dan sumber daya lain secara nyaman dan efisien dibanding proses-proses terpisah.

2.4 Sinkronisasi Proses

Konkurensi merupakan landasan umum perancangan sistem operasi. Proses-proses disebut konkuren jika proses-proses (lebih dari satu proses) berada pada saat yang sama. Konkurensi merupakan landasan umum perancangan sistem operasi. Proses-proses yang konkuren dapat sepenuhnya tak tergantung dengan lainnya tapi dapat juga saling berinteraksi. Proses-proses yang berimteraksi memerlukan sinkronisasi agar terkendali dengan baik(Pangera dan Arriyus, 2008).

Komunikasi antara proses membutuhkan *place by calls* untuk mengirim dan menerima data primitif. Terdapat rancangan yang berbeda-beda dalam implementasi setiap primitif. Pengiriman pesan mungkin dapat diblok (*blocking*) atau tidak dapat dibloking (*nonblocking*) - juga dikenal dengan nama sinkron atau asinkron

Latar belakang dari sinkronisasi dalam sistem operasi adalah:

1. Akses-akses yang dilakukan secara bersamaan ke data yang sama, dapat menyebabkan data menjadi tidak konsisten.
2. Untuk menjaga agar data tetap konsisten, dibutuhkan mekanisme-mekanisme untuk memastikan pemintaan eksekusi dari proses yang bekerja.
3. *Race condition*, yaitu situasi dimana beberapa proses mengakses dan memanipulasi data secara bersamaan. Nilai terakhir dari data bergantung dari proses mana yang selesai terakhir.
4. Untuk menghindari *race condition*, proses-proses secara bersamaan harus disinkronisasikan.

2.4.1 Race Condition

Dalam sistem operasi terdapat *race condition*, yaitu situasi dimana terdapat beberapa proses mengakses dan memanipulasi data yang sama secara bersamaan. Nilai data bergantung dari proses mana yang selesai terakhir. Hal ini akan menyebabkan nilai data menjadi tidak benar, jika tidak terdapat mekanisme untuk mensinkronisasikan data tersebut kepada proses-proses yang menggunakannya. Sebagai contoh, perhatikanlah sebuah print spooler. Ketika sebuah proses ingin mencetak sebuah berkas, proses tersebut memasukkan nama berkas ke dalam sebuah spooler direktori.

Sebuah spooler direktori memiliki slot yang diberi nomor 0, 1, 2, 3, 4,... masing-masing dapat memuat sebuah nama berkas. Terdapat dua variabel, 'Out' menunjuk berkas berikutnya untuk dicetak dan 'In' menunjuk slot kosong di direktori. Berkas pada slot 0, 1, 2, 3 telah selesai dicetak dan slot dikosongkan, dan slot 4, 5, 6 sedang terisi (berisi nama dari berkas yang antre untuk dicetak). Pada saat ini, 'Out' bernilai 4 dan 'In' bernilai 7. Proses A membaca 'In' dan menyimpan nilai 7 di sebuah variabel lokal yang disebut next_free_slot. Sebuah clock interrupt terjadi dan CPU memutuskan bahwa proses A digantikan oleh proses B. Proses B juga membaca 'In', dan juga mengambil nilai 7, sehingga menyimpan nama berkas di slot nomor 7 dan memperbaharui nilai 'In' menjadi 8. Akhirnya proses A berjalan lagi (*restart*), dimulai dari tempat dimana proses tersebut mati. Proses A membaca next_free_slot dan menemukan nilai 7, dan menulis nama berkas di slot nomor 7, menghapus nama berkas yang bau saja diletakkan oleh proses B. Kemudian proses A menghitung next_free_slot + 1, yang nilainya 8 dan memperbaharui nilai 'In' menjadi 8. Direktori spooler sekarang secara internal konsisten, sehingga printer daemon tidak akan memberitahukan apa pun yang terjadi. Proses B tidak akan menghasilkan *output* apa pun. Situasi seperti ini, dimana dua atau lebih proses melakukan proses *reading* atau *writing* beberapa *shared data* dan hasilnya bergantung pada urutan eksekusi disebut *race condition*.

2.4.2 *Critical Section*

Critical Section adalah bagian dari proses yang memerlukan *mutual exclusion*. Suatu sistem terdiri dari n proses di mana semuanya berkompetisi menggunakan data yang di gunakan bersama-sama. Masing masing proses mempunyai sebuah kode segmen yang di sebut dengan *critical section*, di mana sebuah proses memungkinkan untuk mengubah variable umum. Gambaran penting dari sistem adalah ketika sebuah proses di jalankan di dalam *critical section*, tidak ada proses lain yang di izinkan untuk menjalankan *critical section*-nya (Pangera dan Arriyus, 2008). Sebuah solusi dari permasalahan *critical section* harus memenuhi sayarat-syarat sebagai berikut:

1. *Mutual exclusion*, jika suatu proses sedang mengerjakan *critical section*, maka tidak boleh ada proses lain yang masuk (mengerjakan) *critical section* tersebut.
2. *Progress*, jika tidak ada suatu proses yang mengerjakan *critical section* dan ada beberapa proses yang akan masuk ke *critical section*, maka hanya proses-proses yang sedang berada pada *entry-section* saja yang boleh berkompetisi dan mengerjakan *critical section*.
3. *Bounded Waiting*, terdapat batasan jumlah waktu yang diizinkan oleh proses lain untuk memasuki *critical section* setelah sebuah proses membuat permintaan untuk memasuki *critical section*-nya dan sebelum permintaan di kabulkan.

2.4.3 Producer-Consumer Problem

Kasus *producer-consumer* digunakan sebagai ilustrasi pembahasan *sinkronisasi*. Masalah *producer-consumer* disebut juga *bounded-buffer problem* (masalah *buffer* dengan jumlah terbatas). Prinsip dasar dari Masalah *producer-consumer* adalah dua proses atau lebih dapat dapat berkoopersi dengan menggunakan sinyal sederhana sedemikian rupa, sehingga suatu proses dapat di paksa berhenti pada posisi tertentu sampai proses tersebut menerima sinyal tertentu (Stalling, 2003).

Masalah *producer-consumer* dapat dikembangkan menjadi masalah yang memiliki m buah produsen dan n buah konsumen. Karena *buffer* terbatas, masalah berikut dapat terjadi, yaitu:

1. Masalah untuk *producer*.

   Masalah terjadi ketika *buffer* telah penuh, sementara *producer* ingin meletakkan informasi ke *buffer* yang telah penuh itu.

2. Masalah untuk *consumer*.

   Masalah terjadi ketika *consumer* ingin mengambil informasi sementara *buffer* telah/sedang kosong

2.5 Metode *Sleep and Wake up*

Permasalahan dalam multi threading diselesaikan dengan *sinkronisasi*. *Sinkronisasi* akan memutuskan *thread* mana yang mengeksekusi dahulu dan menyimpan informasi eksekusi sehingga ketika tiba gilirannya lagi informasi tersebut dapat dipanggil dan proses eksekusi dapat dilanjutkan. Salah satu metode *sinkronisasi* yang paling sederhana adalah *Sleep and Wake up* (Kumar, 2004). Penyelesaian ini memiliki dua rutin, yaitu *sleep* dan *wakeup*

1. *Sleep* adalah *system call* membuat proses yang memanggil di blok (*blocked*).
2. *Wake up* adalah *system call* yang membuat proses yang memanggil menjadi *ready.*

Mekanismenya proses akan diblok/tidur (*sleep*) apabila tidak bisa memasuki *critical section*-nya dan akan dibangunkan *(wake up)*/*ready* apabila *resource* yang diperlukan telah tersedia. Kedua rutin bersifat *atomic,* yaitu saat rutin dieksekusi maka tak ada interupsi yang dapat menyela.

2.6 Deadlock

Deadlock dapat di definisikan sebagai pemblokiran permanen sejumlah proses yang berkompetisi dalam mendapatkan sumberdaya sistem atau yang berkomunikasi satu dengan yang lainnya (Stalling,2003). Proses disebut *deadlock*

jika proses menunggu satu kejadian tertentu yang tak akan pernah terjadi. Sekumpulan proses berkondisi *deadlock* bila setiap proses yang ada di kumpulan itu menunggu suatu kejadian yang hanya dapat dilakukan proses lain yang juga berada di kumpulan itu. Proses menunggu kejadian yang tidak akan pernah terjadi. *Deadlock* terjadi ketika proses-proses mengakses secara ekslusif sumber daya. Semua *deadlock* yang terjadi melibatkan persaingan memperoleh sumber daya ekslusif oleh dua proses atau lebih.

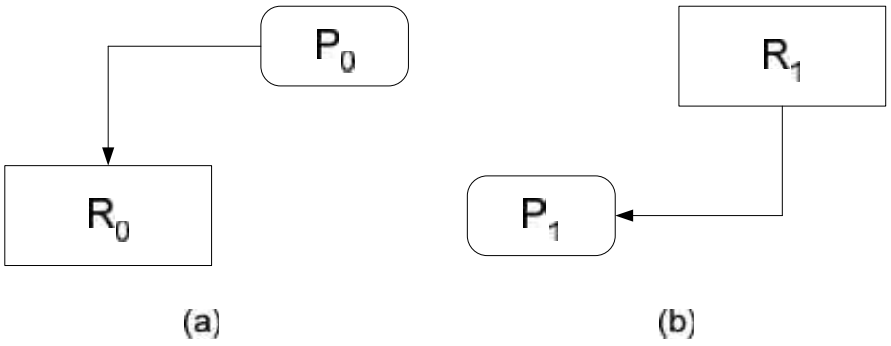
Proses dikatakan sebagai mengalami *starvation* bila proses-proses itu menunggu alokasi sumber daya sampai tak berhingga, sementara proses-proses lain dapat memperoleh alokasi sumber daya. *Starvation* disebabkan bias pada kebijaksanaan atau strategi alokasi sumber daya. Kondisi ini harus dihindari karena tidak adil tetapi dikehendaki penghindaran dilakukan seefisien mungkin.

2.6.1 Model Deadlock

Urutan kejadian pengoperasian perangkat masukan/keluaran adalah:

1. Meminta *(request)* : meminta sumber daya
2. Memakai *(use)* : memakai sumber daya.
3. Melepaskan *(release)* : melepas sumber daya

Misalnya, terdapat dua proses P_0 dan P_1 dan dua sumber daya R_0 dan R_1.



Gambar 2.4 *Graph* meminta sumber daya dan alokasi sumber daya

Gambar 2.2 (a) P_0 meminta sumber daya R_0, ditandai busur (*edge*) berarah dari proses P_0 ke sumber daya R_0. Gambar 2.2 (b) sumber daya R_1 dialokasikan ke

P_1, ditandai busur berarah dari sumber daya R_1 ke proses P_1. Kemudian terjadi skenario berikut,

- P_0 sambil masih menggenggam R_0, meminta R_1.
- P_1 sambil masih menggenggam R_1, meminta R_0.

Kejadian ini mengakibatkan *deadlock* karena sama-sama akan saling menunggu. *Deadlock* tidak hanya terjadi pada dua proses dan dua sumber daya, *deadlock* dpat terjadi dengan melibatkan lebih dari dua proses dan dua sumber daya.

2.6.2 Metode – Metode Mengatasi Deadlock

Terdapat empat syarat untuk mencegah terjadinya *deadlock*, yaitu:

1. *Mutual Exclution Condition*

   Tiap sumber daya saat itu diberikan pada tepat satu proses.
2. Kondisi genggam dan tunggu (*hold and wait condition*)

   Proses-proses yang sedang mengenggam sumber daya, menunggu sumber daya-sumber daya baru.
3. Kondisi *non-preemption* (*non-preemption condition*)

   Sumber daya-sumber daya yang sebelumnya diberikan tidak dapat diambil paksa dari proses itu. Sumber daya-sumber daya harus secara eksplisit dilepaskan dari proses yang mengenggamnya.
4. Kondisi menunggu secara sirkuler (*circular wait condition*)

   Harus terdapat rantai sirkuler dari dua proses atau lebih, masing-masing menunggu sumber daya yang digenggam oleh anggota berikutnya pada rantai itu.

Ketiga syarat pertama merupakan syarat perlu (*necessary conditions*) bagi terjadinya *deadlock*. Keberadaan *deadlock* selalu berarti terpenuhi kondisi-kondisi di atas, tak mungkin terjadi *deadlock* bila tidak ada ketiga kondisi itu. *Deadlock* terjadi berarti terdapat ketiga kondisi itu, tetapi adanya ketiga kondisi itu belum berarti terjadi *deadlock*. *Deadlock* baru benar-benar terjadi bila syarat keempat terpenuhi. Kondisi keempat merupakan keharusan bagi terjadinya peristiwa *deadlock*. Bila salah satu kondisi saja tidak terpenuhi, maka *deadlock* tidak terjadi.

2.6.3 Pencegahan Deadlock

Havender, J.W dalam bukunya berjudul '*Avoiding Deadlock in Multitasking Systems*' mengemukakan jika sembarang syarat dari keempat syarat tak terpenuhi maka tidak akan terjadi *deadlock*. Havender menyarankan starategi-strategi berikut untuk meniadakan syarat-syarat tersebut, yaitu:

1. Tiap proses harus meminta sumber daya yang diperlukan sekaligus dan tidak berlanjut sampai semuanya diberikan.
2. Jika proses telah sedang memegang sumber daya tertentu, untuk permintaan berikutnya proses harus melepas dulu sumber daya yang dipegangnya. Jika diperlukan, proses meminta kembali sekaligus dengan sumber daya yang baru.
3. Beri pengurutan linear terhadap tipe-tipe sumber daya pada semua proses, yaitu jika proses telah dialokasikan suatu tipe sumber daya, proses hanya boleh meminta sumber daya-sumber daya tipe pada urutan yang berikutnya.

2.7 Model dan Simulasi

2.7.1 Model

Model adalah contoh sederhana dari sistem dan menyerupai sifat-sifat sistem yang dipertimbangkan, tetapi tidak sama dengan sistem. Penyederhanaan dari system sangat penting agar dapat dipelajari secara seksama. Model dikembangkan dengan tujuan untuk studi tingkah-laku sistem melalui analisis rinci akan komponen atau unsur dan proses utama yang menyusun sistem dan interaksinya antara satu dengan yang lain.

Model juga dapat didefenisikan sebagai suatu gambaran, abstraksi atau imajinasi suatu sistem nyata. Ataupun berupa suatu abstraksi dunia nyata yang pada akhirnya dapat digunakan untuk mengambil keputusan. Model berisi informasi-informasi tentang sesuatu yang dibuat dengan tujuan untuk mempelajari sistem yang sebenarnya.

Model didefinisikan sebagai suatu deskripsi logis tentang bagaimana sistem bekerja atau komponen-komponen berinteraksi. Dengan membuat model

dari suatu sistem maka diharapkan dapat lebih mudah untuk melakukan analisis. Hal ini merupakan prinsip pemodelan, yaitu bahwa pemodelan bertujuan untuk mempermudah analisis dan pengembangannya (Law and Kelton, 1991).

Model dapat berupa tiruan dari suatu benda, sistem atau peristiwa sesungguhnya yang hanya mengandung informasi-informasi yang dipandang penting untuk ditelaah. Model yang dibuat dapat berguna untuk:

1. Membantu dalam berpikir, model menyajikan deskripsi yang sistematis tentang suatu sistem sehingga dapat mempermudah mempelajari sistem tersebut.
2. Membantu untuk berkomunikasi atau mempermudah menjelaskan tentang suatu sistem kepada orang lain.
3. Sebagai alat latihan, untuk melatih ketrampilan orang-orang yang berhubungan dengan sistem sebenarnya yang dimodelkan. Contohnya: Simulator dalam dunia penerbangan, ini digunakan untuk melatih seorang calon pilot yang dalam taraf belajar, belum boleh mengemudikan pesawat yang sebenarnya, tetapi belajar mengemudikan suatu model yang mewakili pesawat dan juga mengoperasikan model tersebut terhadap suatu model lapangan terbang, udara, lingkungan terbang dan sebagainya.
4. Sebagai alat prediksi terhadap kelakuan sistem untuk waktu yang akan datang, yaitu pengaruh-pengaruh yang ingin diketahui jika ada perubahan sistem atau operasi sistem dan membantu dalam melakukan percobaan, dalam hal melakukan percobaan atau eksperimen tidak mungkin langsung dilaksanakan atau diadakan secara praktis karena biaya yang mahal dan bahaya atau resiko yang tinggi.

Sebelum menentukan model yang akan dibuat, terlebih dahulu dipelajari sistemnya. Sistem yang ada seringkali sangat kompleks, tapi model diusahakan dibuat sesederhana mungkin .

2.7.2 Simulasi

Simulasi adalah tiruan dari proses dunia nyata atau sistem. Simulasi dapat juga di artikan sebagai proses implementasi model menjadi program komputer (*software*) atau rangkaian elektronik dan mengesekusi *software* tersebut

sedemikian rupa sehingga perilakunya menirukan atau menyerupai sistem nyata (realitas) tertentu untuk tujuan mempeajari perilaku sistem, pelatihan atau permainan yang melibatkan sistem nyata.

Ide awal dari simulasi adalah untuk meniru situasi dunia nyata secara matematis, kemudian mempelajari sifat dan karakter operasionalnya, dan akhirnya membuat kesimpulan dan membuat keputusan berdasar hasil dari simulasi. Dengan cara ini, sistem di dunia nyata tidak disentuh /dirubah sampai keuntungan dan kerugian dari apa yang menjadi kebijakan utama suatu keputusan di uji cobakan dalam sistem model.

Studi informatika yang mendukung simulasi komputer, antara lain pemodelan dan simulasi, teori sistem, rekayasa perangkat lunak dan gerak animasi komputer. Proses tahapan dalam mengembangkan simulasi komputer adalah sebagai berikut:

1. Memahami sistem yang akan disimulasikan
2. Mengembangkan model matematika dari system
3. Mengembangkan model matematika untuk simulasi
4. Membuat program (*software*) computer
5. Menguji, memverifikasi dan memvalidasi keluaran simulasi
6. Mengeksekusi program simulasi untuk tujuan tertentu.

Manfaat dari simulasi adalah sebagai berikut :

1. Menjelaskan kelakuan sistem.
2. Menirukan bekerjanya suatu sistem melalui melalui suatu model.
3. Memecahkan suatu persoalan matematik dengan analisis numerik.
4. Mempelajari dinamika suatu sistem.
5. Memberikan suatu deskripsi perilaku sistem dalam perkembangan sejalan dengan bertambahnya waktu.
6. Membangun teori atau hipotesa yang mempertanggungjawabkan kelakuan dari sistem yang diamati.

2.8 Perancangan Sistem

2.8.1 Alat Perancangan Sistem

Alat-alat perancangan alur kerja simulasi menggunakan *State Transition Diag*ram (STD). Ada tiga alasan untuk menggunakan alat perancangan sistem sebelum membuat suatu sistem yaitu:

1. Agar kita bisa fokus pada bagian sistem yang penting.
2. Agar bisa berdiskusi mengenai perubahan-perubahan dan koreksi sesuai keinginan pemakai.
3. Untuk meyakinkan bahwa kita mengerti akan lingkungan pemakai dan memiliki dokumentasi perancangan sistem sehingga programmer bisa membuat sistem tersebut.

2.8.2 *State Transition Diagram*/STD

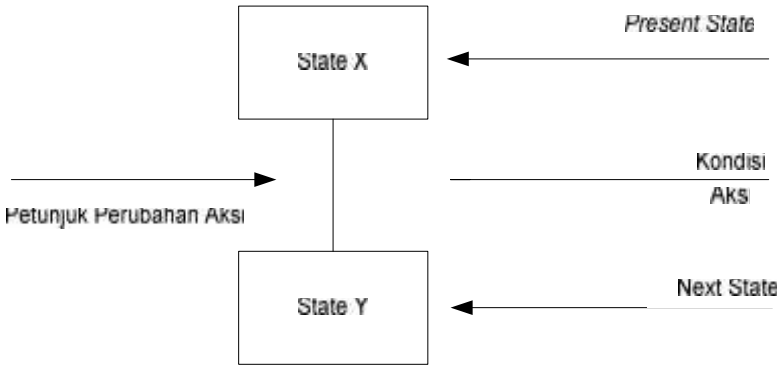
STD adalah sebuah model tingkah laku yang bertumpu pada definisi dari serangkaian keadaan sistem (Pressman,2002).

Cara kerja sistem pada hakikatnya terbagi menjadi dua bagian:

1. Pasif. Sistem tidak melakukan kontrol terhadap lingkungan tetapi lebih bersifat memberikan reaksi.
2. Aktif. Sistem melakukan kontrol terhadap lingkungan secara aktif. Sistem ini sanggup menerima sumber daya eksternal dengan kecepatan tinggi dan dalam waktu singkat (*real time*) memberikan respon terhadap lingkungan sesuai dengan program yang telah ditentukan.

Ada dua pendekatan dalam membuat STD, yaitu:

1. Identifikasi setiap kemungkinan *state* dari sistem dan gambarkan masing-masing pada *state* sebuah kotak, kemudian tentukan hubungan antar *state* tersebut.
2. Dimulai dengan *state* P1 dan dilanjutkan dengan *state* P2, berikutnya dilanjutkan sesuai *flow* yang diinginkan.



Gambar 2.5 Pendekatan Untuk Membuat STD (Pressman, 2002)

Notasi State Transition Diagram (STD)

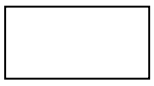
Notasi STD terdiri dari *state* dan *transition state*. *State* adalah kumpulan keadaan atau atribut yang mencirikan seseorang atau suatu benda pada waktu tertentu. Bentuk *state* dibagi menjadi dua, yaitu *Initial State* dan *Final State*. *Initial state* menyatakan awal dari suatu *state* (hanya ada satu state), sedang *Final State* menyatakan aktif dari suatu *state* (bisa lebih dari satu *state*).

Transition State terdiri dari kondisi dan aksi. Kondisi adalah suatu kejadian pada lingkaran luar yang dapat dideteksi oleh sistem. Sedangkan aksi adalah yang dilakukan oleh sistem bila terjadi perubahan *state* atau merupakan reaksi terhadap kondisi.

1. Keadaan sistem

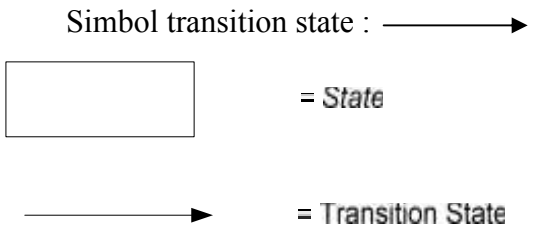
Setiap kotak mewakili suatu keadaan dimana sistem mungkin berada didalamnya. State disimbolkan dengan segi empat.

Simbol state:



2. Perubahan sistem

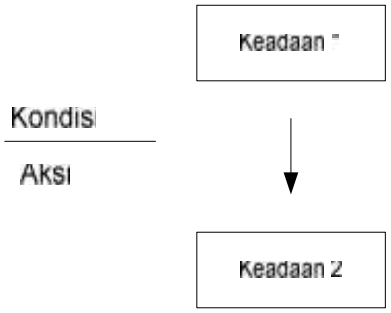
Untuk menghubungkan suatu keadaan dengan keadaan lain, digunakan ini jika sistem memiliki transisi dalam prilakunya, maka hanya suatu keadaan dapat berubah menjadi keadaan tertentu.



Gambar 2.6 Notasi STD

3. Kondisi dan aksi

Untuk melangkapi STD, dibutuhkan dua hal tambahan: kondisi sebelum keadaan berubah dan aksi dari pemakai untuk mengubah keadaan. Dibawah ini adalah ilustrasi dari kondisi dan aksi yang ditampilkan disebelah anak panah yang menghubungkan dua keadaan.

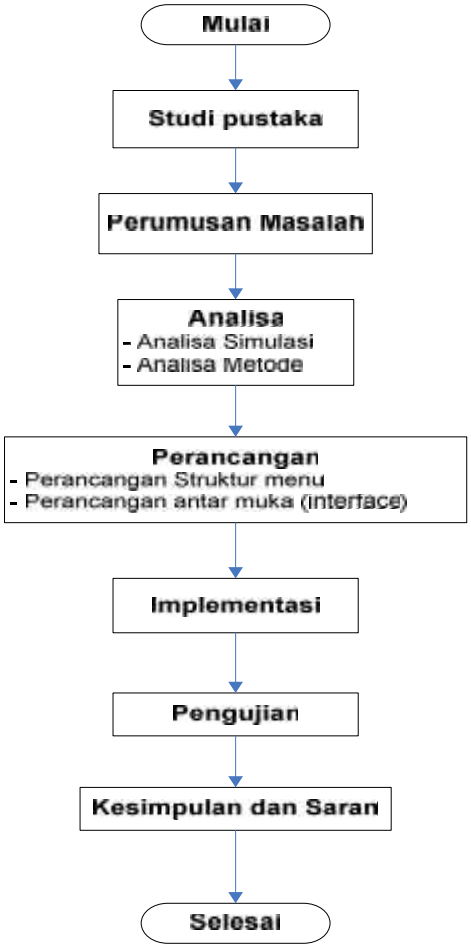


Gambar 2.7 Kondisi dan Aksi

BAB III
METODOLOGI PENELITIAN

Dalam penelitian Tugas Akhir ini, metodologi penelitian merupakan pedoman dalam pelaksanaan penelitian sehingga yang dicapai tidak menyimpang dari tujuan yang telah ditentukan sebelumnya.

Dalam metodologi penelitian dijabarkan tahapan-tahapan yang dilakukan dalam penelitian. Metodologi penelitian terdiri dari beberapa tahapan yang terkait secara sistematis. Tahapan ini diperlukan untuk memudahkan dalam melakukan penelitian. Tahapan yang dilakukan dalam penelitian adalah sebagai berikut:



Gambar 3.1 Alur kerja perancangan Simulasi *Producer-Consumer Problem*

3.1 Studi Pustaka

Pada tahap ini dilakukan proses pengumpulan informasi yang diperlukan untuk proses perencanaan simulasi. Studi pustaka ini mencakup tentang pemahaman konsep. Konsep-konsep tersebut antara lain:

1. Mengenai konsep dari Sistem Operasi
2. Pemahaman tentang kasus *Producer-consumer problem*
3. Pemahaman tantang pemodelan dan simulasi
4. Metode yang di gunakan yaitu Sleep and wake up

3.2 Perumusan Masalah

Setelah informasi didapat, maka pada tahap perumusan masalah ini dirumuskan permasalahan yang akan dipecahkan, yaitu bagaimana merancang suatu perangkat lunak yang dapat memodelkan kasus *producer-consumer problem* dalam bentuk simulasi.

3.3 Analisa

Pada tahap ini dilakukan analisa permasalahan yang telah dirumuskan, yaitu menganalisa kasus *producer-consumer problem* yang digunakan sebagai ilustrasi pembahasan *sinkronisasi*. Kemudian menganalisa kebutuhan simulasi dan menganalisa metode yang akan digunakan.

3.3.1 Analisa Simulasi

Kasus *producer-consumer problem* di simulasikan sebagai sebuah pasar, dimana terdapat *producer, consumer* dan market. *Producer* adalah variable yang memasukkan barang ke market, sementara *consumer* adalah variable yang membeli barang dari market. Analoginya dengan sistem operasi adalah sebagai berikut:

1. *Producer* di analogikan sebagai proses yang meletakkan informasi ke *buffer*
2. *Consumer* di analogikan sebagai proses yang menghapus informasi dari *buffer*

3. Market di analogikan sebagai *buffer*, sumber daya dengan jumlah terbatas yang di akses oleh beberapa proses. Jenis barang pada market mewakili jumlah slot yang ada pada buffer, sedangkan angka pada buffer di analogikan sebagai informasi yang tersimpan di dalam *buffer*
4. Variable pelengkap lainnya, seperti: gambar rumah producer dan consumer, dan beberapa gambar lainnya.

3.3.2 Analisa metode

Mekanismenya proses akan di blok/tidur (sleep) apabila tidak bisa memasuki *critical section*-nya dan akan dibangunkan *(wake up)*/*ready* apabila resource yang diperlukan telah tersedia.

Solusi penyelesaian dengan *sleep and wakeup* adalah sebagai berikut:

1. Solusi untuk masalah *producer*.

Producer memanggil *sleep* begitu mengetahui *buffer* telah penuh saat *producer* akan menyimpan informasi ke *buffer*. *Producer* tidak lagi aktif kecuali dibangunkan (*wake-up*), proses lain (*consumer*) yang memberitahu bahwa satu *barang* atau lebih telah diambil dari *buffer* sehingga terdapat ruang bagi *producer* untuk menyimpan informasi ke *buffer*.

2. Solusi untuk masalah *consumer*.

Consumer memanggil *sleep* begitu mengetahui *buffer* telah kosong saat *consumer* mengambil *barang*. *Consumer* tidak lagi aktif kecuali dibangunkan (*wakeup*) proses lain (*producer*) yang memberitahu bahwa *buffer* telah terisi satu *barang* atau lebih sehingga terdapat informasi yang dapat diambil *consumer* dari *buffer*.

3.4 Perancangan

3.3.1 Perancangan Struktur Menu

Rancangan struktur menu diperlukan untuk memberikan gambaran terhadap menu-menu atau fitur pada sistem yang akan dibangun.

Beberapa komponen yang dipakai dalam membangun struktur menu pada simulasi ini antara lain:

1. *Picturebox*, untuk menampilkan gambar.

2. *Combobox*, untuk memilih salah satu pilihan.
3. *Textbox*, untuk memasukkan *input*.
4. *CommandButton*, sebagai tombol.
5. *Frame,* untuk mengelompokkan objek.
6. *Timer*, komponen untuk menjalankan proses animasi (memeriksa keadaan dan menggerakkan keadaan ke keadaan berikutnya).

3.3.2 Perancangan Antar Muka (*Interface)*

Untuk mempermudah komunikasi antara sistem dengan pengguna, maka perlu dirancang antar muka *(interface)*. Dalam perancangan *interface* hal terpenting yang ditekankan adalah bagaimana menciptakan tampilan yang baik dan mudah dimengerti oleh pengguna.

3.5 Implementasi

Implementasi merupakan kelanjutan dari tahap perancangan sistem yang telah didesain. Implementasi juga merupakan tahap pembangunan sistem menggunakan perangkat keras dan perangkat lunak yang telah ditetapkan.

Tujuan implementasi antara lain:

1. Menyelesaikan desain sistem yang ada dalam perancangan yang telah di buat.
2. Menguji dan mendokumentasikan program-program atau prosedur-prosedur dari perancangan sistem yang telah di buat.
3. Memastikan bahwa pemakai dapat mengoperasikan sistem.
4. Mempertimbangkan bahwa sistem memenuhi permintaaan pemakai yakni dengan menguji secara keseluruhan.
5. Memastikan bahwa sistem berjalan dengan benar.

3.6 Pengujian

Pada tahapan pengujian (*testing*) ini menggambarkan kondisi-kondisi yang terjadi apabila simulasi dijalankan. Apakah sistem telah sesuai dengan kebutuhan atau tidak yaitu dengan cara menguji fitur-fitur yang ada, *interface*, dan

performance yang mencakup seluruh aspek dari sistem yang dibangun. Pengujian dilakukan dengan menggunakan metode pengujian *blackbox*.

3.6 Kesimpulan dan Saran

Kesimpulan dan saran merupakan tahap akhir dari penelitian yang dilakukan. Dibagian ini akan ditarik kesimpulan berdasarkan hasil dari penelitian serta memberikan saran-saran untuk menyempurnakan dan mengembangkan penelitian itu.

BAB V
IMPLEMENTASI DAN PENGUJIAN

Implementasi dan pengujian merupakan tahap yang dilakukan setelah tahap analisa dan perancangan selesai dikerjakan.

5. 1 Implementasi

Implementasi merupakan kelanjutan dari tahap perancangan sistem yang telah didesain. Implementasi juga merupakan tahap pembangunan sistem menggunakan perangkat keras dan perangkat lunak yang telah ditetapkan.

Tujuan implementasi antara lain:

1. Menyelesaikan desain sistem yang ada dalam perancangan yang telah di buat.
2. Menguji dan mendokumentasikan program-program atau prosedur-prosedur dari perancangan sistem yang telah di buat.
3. Memastikan bahwa pemakai dapat mengoperasikan sistem.
4. Mempertimbangkan bahwa sistem memenuhi permintaaan pemakai yakni dengan menguji secara keseluruhan.
5. Memastikan bahwa sistem berjalan dengan benar.

5.1. 1 Lingkungan Implementasi

Lingkungan implementasi sistem ada 2 (dua) yaitu: lingkungan perangkat keras dan lingkungan perangkat lunak.

5.1.1.1 Perangkat Keras

Perangkat keras yang digunakan dalam Simulasi Simulasi Producer-Consumer Problem ini menggunakan komputer dengan spesifikasi:

1. *Processor* : Intel Core 2 Duo 1.73 GHz
2. *Memory* : 2048 MB
3. *Harddisk* : 120 GB

5.1.1.2 Perangkat lunak

Perangkat lunak yang digunakan dalam impelementasi ini menggunakan:

1. Sistem Operasi : *Windows XP*
2. Pemrograman : *Visual Basic 6.0*
3. Rancangan Tampilan : *Adobe Photoshop CS*

5.1.2 Hasil Implementasi

5.1.2.1 Halaman Splash Screen

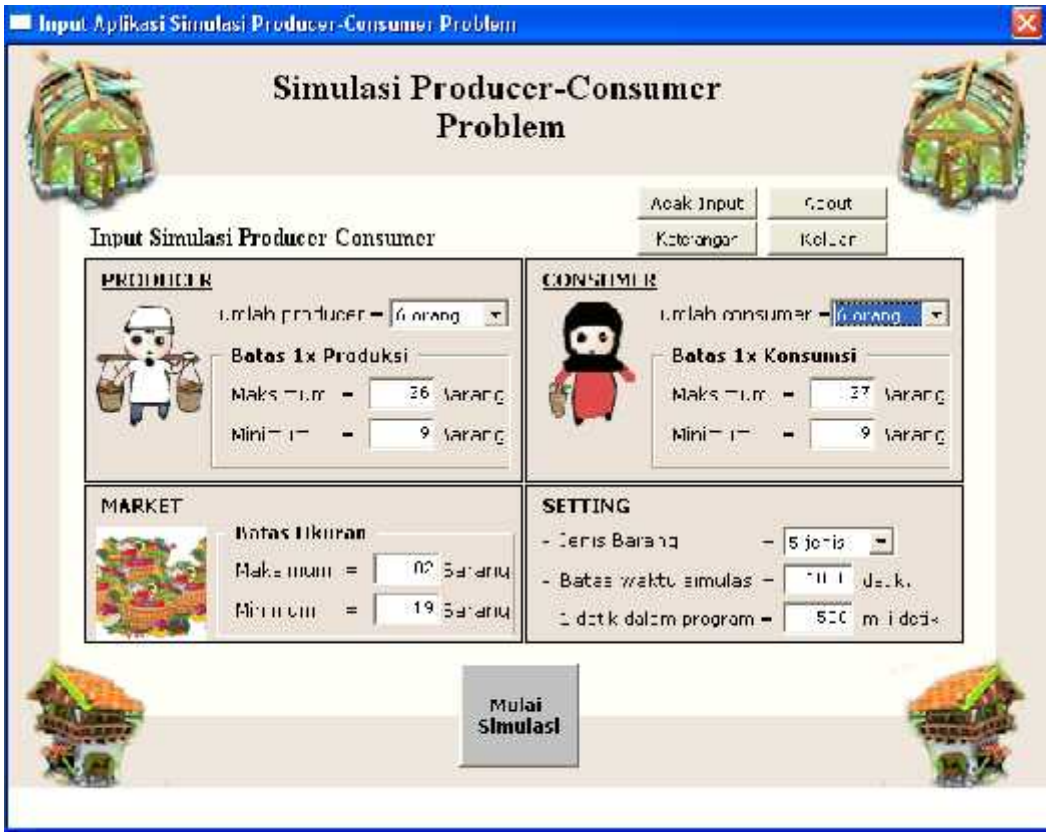
Hasil Implementasi menampilkan halaman *splash screen* berisi judul dan dan nama pembuat perangkat lunak. Tampilan ini tampil pertama kali ketika perangkat lunak dijalankan. Tampilan ini dimaksudkan untuk memperkenalkan perangkat lunak secara singkat ketika dijalankan.

Gambar 5.1 Tampilan Splash Screen

5.1.2.2 Halaman Input

Halaman *input* berfungsi untuk memasukkan *input* perangkat lunak. *Input* perangkat lunak berupa jumlah *producer*, jumlah *customer*, batas maksimum dan minimum bagi *producer* dalam satu kali produksi, batas maksimum dan minimum bagi *consumer* dalam satu kali konsumsi, batas ukuran maksimum dan minimum

market, banyak jenis barang dan kecepatan proses simulasi. Dalam halaman terdapat beberapa tombol menu, yaitu menu *simulasi, about, acak input* dan *keluar*. menu *simulasi* berfungsi menampilkan halaman *Simulasi*. menu *about* berfungsi untuk menampilkan halaman *about*. menu acak input berfungsi untuk mengacak nilai inputan dari Jumlah *producer*, *consumer*, batas maksimum dam minimum dari producer dan consumer, batas ukuran market*(buffer)* dan jenis barang, serta menu keluar berfungsi untuk keluar dari *Simulasi Producer-Consumer Problem.*



Gambar 5.2 Tampilan Input

5.1.2.3 Halaman Simulasi

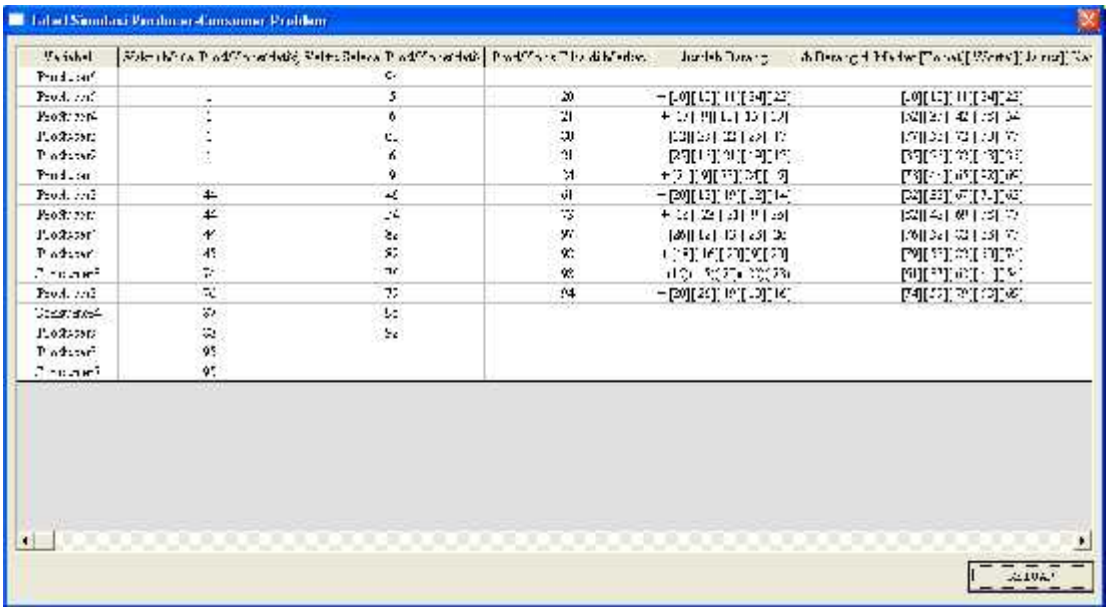
Halaman Simulasi berfungsi untuk menampilkan proses simulasi *Producer-Consumer Problem (bounded buffer problem)*. Dalam halaman ini ini, terdapat enam buah pintu keluar *producer*, 6 buah pintu keluar *consumer* dan sebuah *market*

(illustrasi dari *buffer*), sebagai tempat untuk meletakkan dan mengambil *barang*. Terdapat lima tombol menu pada halaman ini yaitu menu *mulai, hentikan, history, tabel dan keluar*. Menu mulai berfungsi untuk memulai simulasi. Menu hentikan berfungsi untuk menghentikan jalannya simulasi dan simulasi dapat berjalan kembali apabila *user* menekan menu mulai. Menu history berfungsi menampilkan halaman history dan menu tabel berfungsi untuk menampilkan halaman tabel. Tombol menu keluar berfungsi untuk keluar dari tampilan simulasi dan kembali pada tampilan *input.*

Gambar 5.3 Tampilan Simulasi

5.1.2.4 Halaman Tabel Simulasi

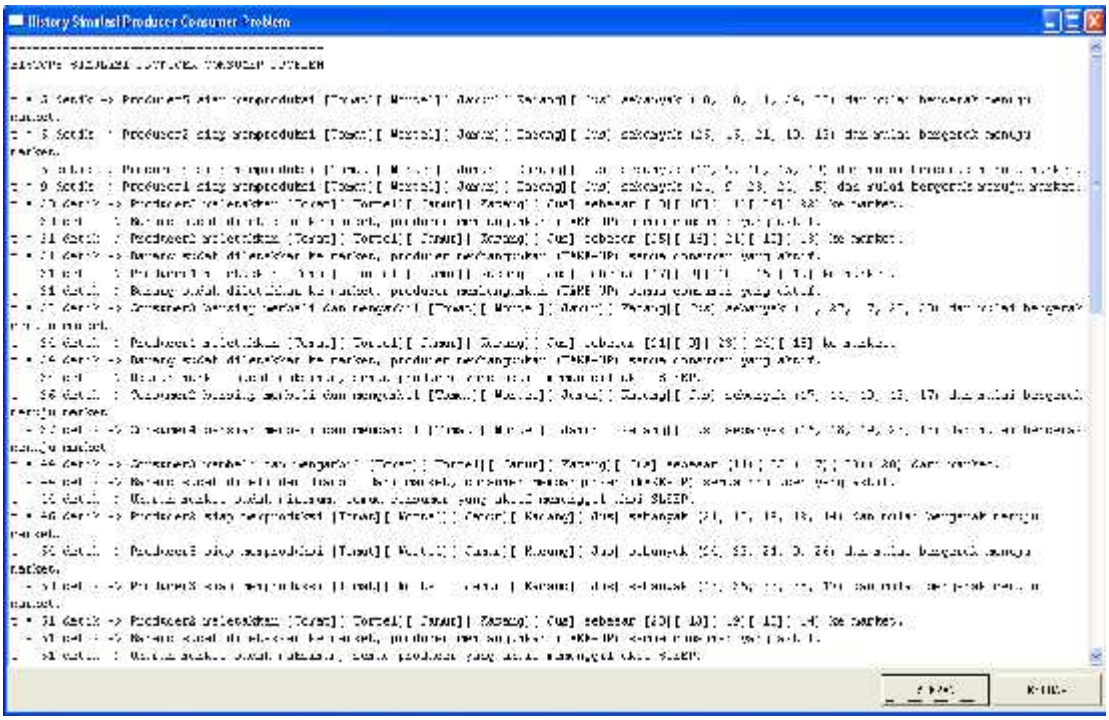
Halaman Tabel Simulasi berfungsi untuk menampilkan tabel laporan yang mencatat setiap waktu mulai produksi/konsumsi, waktu selesai produksi/konsumsi, waktu tiba di *market* dan penambahan serta pengurangan *barang* dalam *market* selama proses simulasi berlangsung.



Gambar 5.4 Tampilan Tabel Simulasi

5.1.2.5 Halaman History

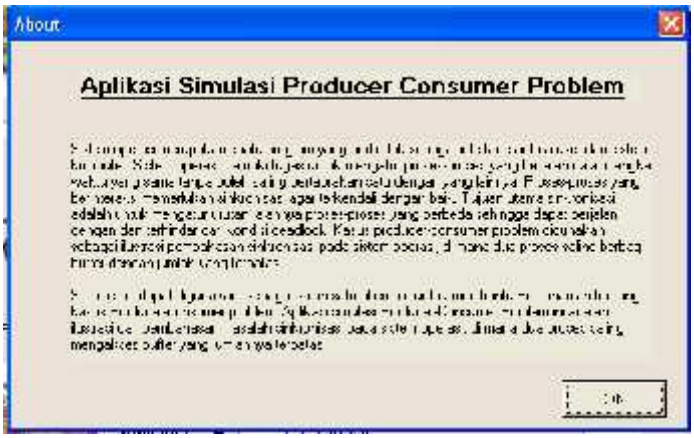
Halaman *History* berfungsi untuk mencatat dan menampilkan laporan proses yang telah terjadi pada proses simulasi. Laporan proses berupa waktu transaksi dan jenis transaksi yang terjadi (penyimpanan barang ke *market*, pengambilan barang ke *market*, aksi *sleep* dan *wake-up* yang dipanggil oleh *producer* atau *consumer*).



Gambar 5.5 Tampilan History

5.1.2.6 Halaman About

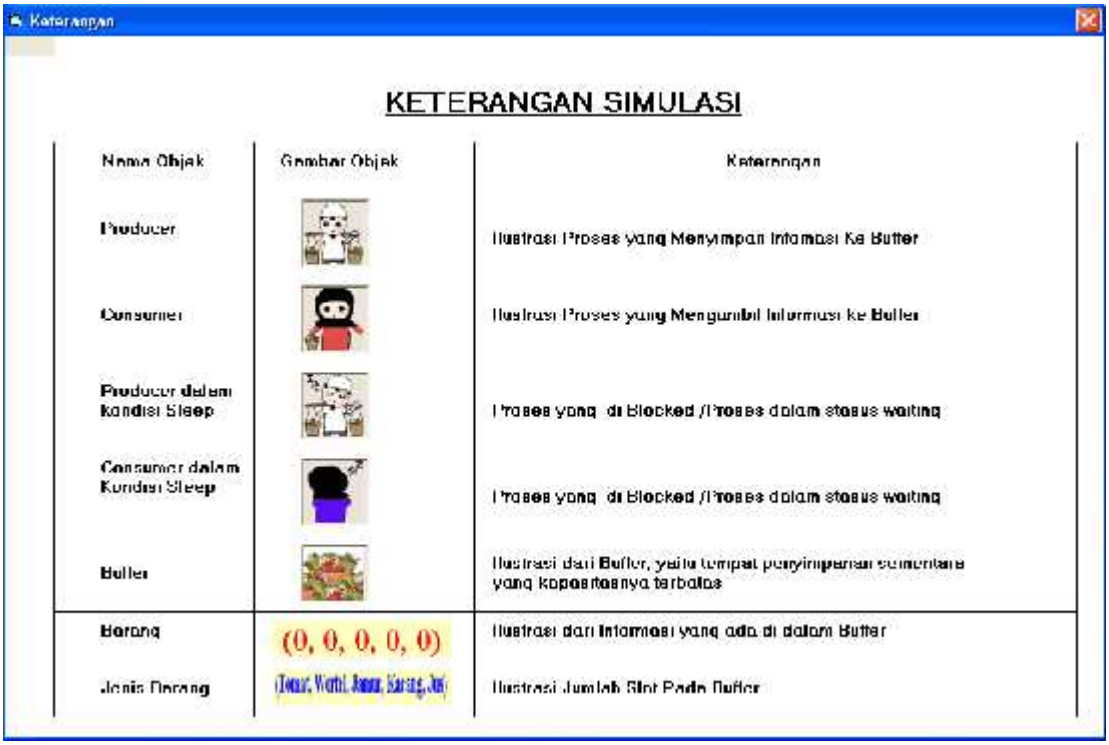
Halaman *About* berfungsi untuk menampilkan judul/nama perangkat lunak, dan keterangan tentang Simulasi *Producer-Consumer Problem.*



Gambar 5.6 Tampilan About

5.1.2.7 Halaman Keterangan Simulasi

Halaman *Keterangan Simulasi* berfungsi untuk melihat keterangan model yang mewakili dari keadaan yang sebenarnya.



Gambar 5.7 Tampilan Halaman Keterangan Simulasi

5.2 Pengujian Sistem

Setelah tahap implementasi dilakukan maka dilanjutkan dengan pengujian dari implementasi yang telah dibuat. Tahap pengujian diperlukan agar dapat diketahui hasil dari implementasi simulasi sesuai dengan analisa yang telah di lakukan. Simulasi ini diuji dengan menggunakan metode pengujian *black box.* Metode *black box* yaitu metode pengujian yang menguji suatu sistem tanpa harus mengetahui proses internal yang berada pada sistem tersebut. Berikut ini akan dijelaskan tahap-tahap pengujian terhadap simulasi yang dibuat.

5.2.1 Pengujian Tampilan

5.2.1.1 Pengujian Tampilan Splash screen

Tabel 5.1 Tabel Pengujian Splash Screen

No	Deskripsi	Prekondisi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian Tampilan Masuk	Pengujian Tampilan layar *Splash Screen*	Klik menu masuk	Klik Menu Tabel	Muncul Tampilan Splash Screen	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil masuk ke layar utama

5.2.1.2 Pengujian Halaman Input

Tabel 5.2 Tabel Pengujian Halaman Input

No	Deskripsi	Prekondisi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian Tampilan Menu Input	Pengujian Tampilan layar menu Input	Klik menu masuk	Klik Menu masuk	Muncul Padalayar Menu input tombol-tombol menu pilihanyaitu menu simulasi, about, acak input, keluar	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil masuk ke layar utama
				Klik Menu simulasi	Muncul Layar Tampilan Simulasi	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasi Masuk ke tampilan Simulasi

				Klik Menu Acak Input	Inputan Berhasil di Inputkan secara Acak	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasi Masuk ke tampilan Simulasi
				Klik Menu About	Muncul Layar Tampilan About	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasi Masuk ke tampilan Simulasi
2.	Pengujian keluar simulasi (*Exit*)	Hasil tampilan layar menu utama sudah tampil	Klik tombol keluar (X)	Klik menu keluar (*Exit*)	Keluar dari simulasi Simulasi Producer-consumer	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil keluar dari simulasi

5.2.1.3 Pengujian Halaman Simulasi

Tabel 5.3 Tabel Pengujian Halaman Simulasi

No	Deskripsi	Prekondisi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian Tampilan Simulasi	Pengujian Tampilan layar Simulasi	Klik menu Simulasi	Klik Menu Simulasi	Muncul Padalayar Tampilan Simulasi dan Menu Pilihan, yaitu menu Mulai, Hentikan, History, Tabel, Keluar	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil masuk ke layar utama
				Klik Menu Mulai	Simulasi berjalan pada Tampilan Simulasi	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasi Masuk ke tampilan Simulasi

				Klik Menu Hentikan	Simulasi yang sedang berjalan berhenti	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasi Masuk ke tampilan Simulasi
				Klik Menu History	Muncul Layar Tampilan History	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasi Masuk ke tampilan Simulasi
				Klik Menu Tabel	Muncul Layar Tampilan Tabel	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasi Masuk ke tampilan Simulasi
2.	Pengujian keluar simulasi (*Exit*)	Hasil tampilan layar menu utama sudah tampil	Klik tombol keluar	Klik menu keluar	Keluar dari Tampilan Simulasi.	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil keluar dari Tampilan Simulasi

5.2.1.4 Pengujian Tampilan History

Tabel 5.4 Tabel Pengujian Halaman History

No	Deskripsi	Prekondisi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian Tampilan History	Pengujian Tampilan layar History	Klik menu History	Klik Menu History	Muncul Pada layar History tombol-tombol menu pilihanyaitu menu Simpan dan Keluar	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil Masuk Ke Menu History
				Klik menu History	Muncul Halaman Untuk	Layar yang ditampilkan sesuai	Berhasil Masuk Ke Menu

					Menyimpan Dokumen	dengan yang diharapkan	History
2.	Pengujian keluar simulasi (*Exit*)	Hasil tampilan layar menu utama sudah tampil	Klik tombol keluar (X)	Klik menu keluar (*Exit*)	Keluar dari Tampilan History	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil keluar dari Tampilan History

5.2.1.5 Pengujian Halaman Tabel Simulasi

Tabel 5.5 Tabel Pengujian Halaman Tabel

No	Deskripsi	Prekondisi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian Tampilan Tabel	Pengujian Tampilan layar Tabel	Klik menu Tabel	Klik Menu Tabel	Muncul Data Jalannya Simulasi dalam Bentu Tabel dan menu Keluar	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil Masuk Ke Menu History
2.	Pengujian keluar simulasi (*Exit*)	Hasil tampilan layar menu utama sudah tampil	Klik tombol keluar (X)	Klik menu keluar (*Exit*)	Keluar dari Tampilan Tabel	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil keluar dari Tampilan Tabel

5.2.1.6 Pengujian Halaman About

Tabel 5.6 Tabel Pengujian Halaman About

No	Deskripsi	Prekondisi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian Tampilan About	Pengujian Tampilan layar About	Klik menu About	Klik Menu About	Menampilan Tampilan about	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil Masuk Ke Tampilan About
2.	Pengujian keluar simulasi (*Exit*)	Hasil tampilan layar menu utama sudah tampil	Klik tombol keluar (X)	Klik menu OK (*Exit*)	Keluar dari Tampilan About	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil keluar dari Tampilan About

5.2.1.7 Pengujian Halaman Keterangan

Tabel 5.7 Tabel Pengujian Halaman Keterangan

No	Deskripsi	Prekondisi	Prosedur pengujian	Masukan	Keluaran yang diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian Tampilan ketera ngan	Pengujian Tampilan layar Ketera ngan	Klik menu keterangan	Klik Menu ketera ngan	Menampilan Tampilan keterangan	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil Masuk Ke Tampilan keterangan
2.	Pengujian keluar simulasi (*Exit*)	Hasil tampilan layar menu utama sudah tampil	Klik tombol keluar (X)	Klik menu OK (*Exit*)	Keluar dari Tampilan keterangan	Layar yang ditampilkan sesuai dengan yang diharapkan	Berhasil keluar dari Tampilan keterangan

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Kesimpulan dari hasil penelitian tugas akhir ini adalah:

1. Simulasi *Producer-Consumer Problem* ini adalah ilustrasi dari pembahasan masalah *sinkronisasi* pada sistem operasi, di ketika dua proses saling mengakses *buffer* yang jumlahnya terbatas
2. Simulasi ini dapat digunakan sebagai salah satu alternatif untuk membantu pemahaman tentang kasus *Producer-consumer problem*.

6.2 Saran

Saran yang perlu diperhatikan untuk pengembangan simulasi lebih lanjut di masa yang akan datang adalah:

1. Simulasi dapat dikembangkan dengan menampilkan halaman *history* pada saat simulasi sedang berjalan.
2. Perangkat lunak dapat dikembangkan dengan menambahkan animasi *producer* dan *consumer* yang lebih detil, seperti animasi *producer* sedang memproduksi *barang*, animasi *consumer* sedang mengkonsumsi *barang*.

DAFTAR PUSTAKA

Hariyanto.B, *Sistem Operasi* , Informatika, Bandung, 2009.

Kusnadi, Kuswoyo. A, dan Sigit Purnomo*,"Sistem Operasi",* Andi Yogyakarta, 2008

Law, A.M, Kelton, W.D, *Simulatin Modeling and Analysis*, McGraw-Hill, Inc, USA, 1991

Pangera, Ariyus, *Sistem Operasi*, Andi, Yogyakarta, 2008.

Roger S. Pressman, *Rekayasa Perangkat Lunak Pendekatan Praktisi (Buku Satu),* ANDI Yogyakarta, 2002.

Shelly, Cashman , *Menjelajah Dunia Komputer Fundamental*, Penerbit Salemba infotek, 2010

Sri Kusuma Dewi,"system Operasi", edisi 2, Graha Ilmu, Jakarta, 2000

Stalling William, *Sistem Operasi dan Prinsip-prinsip Perancangan*, 2003

Sudhir Kumar, *Encyclopaedia of Operating System*, Anmol Publications PVT. LTD, 2004